

БИБЛИОТЕЧКА «КВАНТ»
выпуск 30

М. Н. АРШИНОВ
Л. Е. САДОВСКИЙ

КОДЫ И МАТЕМАТИКА

0101
1010
1100
0011

GF(q)





БИБЛИОТЕЧКА «КВАНТ»
выпуск 30

М. Н. АРШИНОВ
Л. Е. САДОВСКИЙ

КОДЫ И МАТЕМАТИКА

РАССКАЗЫ О КОДИРОВАНИИ



МОСКВА «НАУКА»
ГЛАВНАЯ РЕДАКЦИЯ
ФИЗИКО-МАТЕМАТИЧЕСКОЙ ЛИТЕРАТУРЫ
1983

22.18
А 89
УДК 519.6

М. Н. Аршинов, Л. Е. Садовский

А89 Коды и математика (рассказы о кодировании). — М.: Наука, Главная редакция физико-математической литературы, 1983. — 144 с. (Библиотечка «Квант». Вып. 30). — 25 коп.

В популярной форме книга знакомит читателя с основными понятиями и идеями теории эффективного и помехоустойчивого кодирования — важного направления математики.

Имея своими первоисточниками криптографию (искусство засекречивания истинного содержания сообщения), но главным образом решая различные проблемы, возникающие при передаче информации по линиям связи, теория кодирования в настоящее время выросла в обширную и разветвленную область знания со своим кругом объектов и задач.

Не ставя перед собой цели систематического изложения теории, авторы стремятся отразить главные ее черты.

А $\frac{1702070000-075}{053(02)-83}$ 186-82

ББК 22.18
6Ф0.1

А $\frac{1702070000-075}{053(02)-83}$ 186-82

ПРЕДИСЛОВИЕ

Право же, не будет ошибкой предположить, что у большинства читателей слова «код», «кодирование» вызывают примерно одинаковые представления. Ведь все хорошо знают, что коды или шифры используются для передачи секретной информации. Менее известно, однако, что в наше время коды приобрели и иное значение, быть может, более обыденное, но зато куда более важное и широкое. В этой их новой роли коды и кодирование — прежде всего средство для экономной, удобной и практически безошибочной передачи сообщений. Новые применения кодов сложились в результате бурного развития различных средств связи, неизмеримо возросшего объема передаваемой информации.

Решать возникшие в связи с этим задачи было бы невозможно без привлечения самых разнообразных математических методов. Неслучайно поэтому теория кодирования считается сейчас одним из наиболее важных разделов прикладной математики. Желание познакомить широкий круг читателей с задачами и методами этой теории и является основной нашей целью. Все же немного места уделили мы также кодам в их изначальном смысле — как средству обеспечения секретности.

Первая часть книги (§§ 1—10) написана вполне элементарно, и для ее понимания читателю достаточно ознакомиться с приложением 1, содержащим простейшие сведения о сравнениях.

В дальнейшем изложении, однако, существенно используются основные факты линейной алгебры, а также факты, связанные с понятиями поля и группы. Все необходимые определения и теоремы содержатся в приложениях 2—5.

Не освоившись с материалом этих приложений, читатель не смог бы свободно ориентироваться во второй части книги.

В заключение отметим, что в конце большинства параграфов имеется раздел «Задачи и дополнения», где рассматриваются некоторые более специальные и, как правило, более трудные вопросы, а также приводятся задачи для самостоятельного решения. Читателю, желающему основательно разобраться в содержании книги, мы рекомендуем не пренебрегать этими задачами.

М. Н. Аршинов, Л. Е. Садовский

1. КОДИРОВАНИЕ — ИСТОРИЯ И ПЕРВЫЕ ШАГИ

Коды появились в глубокой древности в виде криптограмм (по-гречески — тайнописи), когда ими пользовались для засекречивания важного сообщения от тех, кому оно не было предназначено. Уже знаменитый греческий историк Геродот (V век до н. э.) приводил примеры писем, понятных лишь для одного адресата. Спартанцы имели специальный механический прибор, при помощи которого важные сообщения можно было писать особым способом, обеспечивающим сохранение тайны. Собственная секретная азбука была у Юлия Цезаря. В средние века и эпоху Возрождения над изобретением тайных шифров трудились многие выдающиеся люди, в их числе философ Фрэнсис Бэкон, крупные математики Франсуа Виет, Джероламо Кардано, Джон Валлис.

С течением времени начали появляться по-настоящему сложные шифры. Один из них, употребляемый и поныне, связан с именем ученого аббата из Вюрцбурга Тритемиуса, которого к занятиям криптографией побуждало, быть может, не только монастырское уединение, но и потребность сохранять от огласки некоторые духовные тайны. Различные хитроумные приемы кодирования применяли шифровальщики при папском дворе и дворах европейских королей. Вместе с искусством шифрования развивалось и искусство дешифровки, или, как говорят, криптоанализа.

Секретные шифры являются неотъемлемой принадлежностью многих детективных романов, в которых действуют изощренные в хитрости шпионы. Писатель-романтик Эдгар По, которого иногда причисляют к создателям детективного жанра, в своем рассказе «Золотой жук» в художественной форме изложил простейшие приемы шифрования и расшифровки сообщений. Эдгар По относился к проблеме расшиф-

ровки оптимистически, вложив в уста своего героя следующую фразу: «...едва ли разуму человека дано загадать такую загадку, которую разум другого его собрата, направленный должным образом, не смог бы раскрыть. Прямо скажу, если текст зашифрован без грубых ошибок и документ в приличной сохранности, я больше ни в чем не нуждаюсь; последующие трудности для меня просто не существуют». Столетие спустя это высказывание было опровергнуто ученым, заложившим основы теории информации, Клодом Шенноном. Шеннон показал, как можно построить криптограмму, которая не поддается никакой расшифровке, если, конечно, не известен способ ее составления.

О некоторых приемах криптографии и криптоанализа мы расскажем в следующем параграфе, в остальных частях книги речь будет идти в основном об ином направлении в кодировании, которое возникло уже в близкую нам эпоху. Связано оно с проблемой передачи сообщений по линиям связи, без которых (т. е. без телеграфа, телефона, радио, телевидения и т. д.) немыслимо наше нынешнее существование. В задачу такого кодирования, как уже говорилось, входит отнюдь не засекречивание сообщений, а иная цель: сделать передачу сообщений быстрой, удобной и надежной. Предназначенное для этой цели кодирующее устройство сопоставляет каждому символу передаваемого текста, а иногда и целым словам или фразам (сообщениям) определенную комбинацию сигналов (приемлемую для передачи по данному каналу связи), называемую кодом или кодовым словом. При этом операцию перевода сообщений в определенные последовательности сигналов называют кодированием, а обратную операцию, восстанавливающую по принятым сигналам (кодовым словам) передаваемые сообщения,— декодированием.

Заметим сразу же, что различные символы или сообщения должны кодироваться различными кодовыми словами, в противном случае по кодовым словам нельзя было бы восстановить передаваемые сообщения.

Исторически первый код, предназначенный для передачи сообщений, связан с именем изобретателя телеграфного аппарата Сэмюэля Морзе и известен всем как азбука Морзе. В этом коде каждой букве или цифре сопоставляется своя последовательность из кратковременных (называемых точками) и длительных (тире) импульсов тока, разделяемых паузами. Другой код, столь же широко распространенный в телеграфии (код Бодо), использует для кодирования два элементарных сигнала — импульс и паузу, при этом со-

поставляемые буквам кодовые слова состоят из пяти таких сигналов.

Коды, использующие два различных элементарных сигнала, называются двоичными. Удобно бывает, отвлекаясь от их физической природы, обозначать эти два сигнала символами 0 и 1. Тогда кодовые слова можно представлять как последовательности из нулей и единиц.

Двоичное кодирование тесно связано с принципом дихотомии (деления пополам). Поясним этот принцип на примере.

Некто задумал число, заключенное между 0 и 7. Угадывающему разрешено задавать вопросы, ответы на которые даются лишь в форме «да» или «нет». Каким образом следует задавать вопросы, чтобы возможно быстрее узнать задуманное число?

Самый бесхитростный путь — перебирать числа в любом порядке, надеясь на удачу. В этом случае при везении может хватить и одного вопроса, но если не повезет, то может понадобиться и целых семь. Поэтому не будем рассчитывать на везение и постараемся построить такую систему вопросов, чтобы любой из ответов — «да» или «нет» — давал нам одинаковую (пусть сначала и неполную) информацию о задуманном числе. Например, первый вопрос может быть таким: «Заключено ли задуманное число в пределах от 0 до 3?» Оба ответа — и «да» и «нет» — одинаково приближают нас к цели: в любом случае остаются четыре возможности для неизвестного числа (а первоначально их было восемь).

Если на первый вопрос получен утвердительный ответ, то во второй раз можно спросить: «Не является ли задуманное число нулем или единицей?»; если же ответ был отрицательным, спросим: «Не является ли задуманное число четверкой или пятеркой»? В любом случае после ответа на второй вопрос останется выбор из двух возможностей. Для того чтобы его осуществить, достаточно одного вопроса. Итак, для угадывания задуманного числа, каким бы оно ни было, достаточно трех вопросов (каждый из них выясняет, содержится ли задуманное число в «нижней» половине заключающего его промежутка). Можно показать, что меньшего числа вопросов недостаточно.

Если возможные ответы «да» или «нет» обозначить условно символами 0 и 1, то ответы запишутся в виде последовательности, состоящей из нулей и единиц. Так, например, если задуманное число было нулем, то на каждый из трех вопросов ответом будет «да». Трем «да» соответствует последовательность 000.

Если было задумано число 3, то ответами будут «да», «нет», «нет», т. е. числу 3 соответствует последовательность 011. По результатам ответов можно составить следующую таблицу:

Таблица 1

Задуманное число	0	1	2	3	4	5	6	7
Ответы	000	001	010	011	100	101	110	111

Читатель, знакомый с двоичной системой счисления, узнает в нижней строке двоичную запись соответствующих чисел верхней строки.

Заметим, что вместо множества чисел от 0 до 7 можно рассматривать любое множество из восьми сообщений, и каждое из них мы можем закодировать последовательностями из нулей и единиц длины 3. Если использовать более длинные двоичные последовательности, то ими в принципе можно закодировать любое конечное множество сообщений.

Действительно, число двоичных последовательностей длины 3 равно $2^3=8$ (все они приведены в таблице 1), двоичных последовательностей длины 4 вдвое больше — число их равно $2^4=16$. Вообще, число двоичных последовательностей длины n равно 2^n. Поэтому, если требуется закодировать нулями и единицами, к примеру, 125 сообщений, то для этого с избытком хватит двоичных последовательностей длины 7 (их в нашем распоряжении имеется $2^7=$ $=128$). Из этого примера становится ясно, что M сообщений можно закодировать двоичными последовательностями длины n тогда и только тогда, когда выполняется условие $2^n \geqslant M$, т. е. когда $n \geqslant \log_2 M$.

Первый, кто понял, что для кодирования достаточно двух символов, был Фрэнсис Бэкон. Двоичный код, который он использовал в криптографических целях, содержал пятиразрядные (как и в коде Бодо) слова, составленные из символов 0, **L**.

Сказанное здесь — это лишь первые подступы к проблеме кодирования, которой посвящена эта книга. Пока же отметим только, что наряду с двоичными кодами применяют коды, использующие не два, а большее число элементарных сигналов, или, как их еще называют, кодовых символов. Их число d называют основанием кода, а множество кодовых

символов — кодовым алфавитом. При этом общее число n-буквенных слов, использующих d символов, вычисляется аналогично прежнему и равно d^n.

Задачи и дополнения

1. Часто по разным соображениям для кодирования сообщений используют не все последовательности в данном алфавите, а только некоторые из них, удовлетворяющие тем или иным ограничениям. Будем рассматривать, например, n-буквенные двоичные слова с фиксированным числом t единиц (или, как говорят, слова постоянного веса t). Сколько всего таких слов — нетрудно подсчитать. Каждое из них получится, если мы выберем некоторым образом t позиций из n, и впишем в них единицы, а в остальных $n-t$ позициях — нули. Значит, число всех слов постоянного веса совпадает с числом сочетаний из n элементов по t, т. е. равно

$$C_n^t = \frac{n!}{t!\,(n-t)!}.$$

2. Сложнее найти число всех двоичных слов длины n, не содержащих несколько нулей подряд. Обозначим это число через s_n. Очевидно, $s_1=2$, а слова длины 2, удовлетворяющие нашему ограничению, таковы: 10, 01, 11, т. е. $s_2=3$. Пусть $\alpha_1\alpha_2\ldots\alpha_{n-1}\alpha_n$ — такое слово из n символов. Если символ $\alpha_n=1$, то $\alpha_1\alpha_2\ldots\alpha_{n-1}$ может быть произвольным $(n-1)$-буквенным словом, не содержащим нескольких нулей подряд. Значит, число слов длины n с единицей на конце равно s_{n-1}.

Если же символ $\alpha_n=0$, то обязательно $\alpha_{n-1}=1$, а первые $n-2$ символа $\alpha_1\alpha_2\ldots\alpha_{n-2}$ могут быть произвольными с учетом рассматриваемого ограничения. Следовательно, имеется s_{n-2} слов длины n с нулем на конце. Таким образом, общее число интересующих нас слов равно

$$s_n = s_{n-1} + s_{n-2}.$$

Из полученного соотношения (подобные соотношения называют рекуррентными) легко можно найти числа s_n для любого n. Поскольку s_1 и s_2 известны, то $s_3=s_1+s_2=5$; $s_4=s_2+s_3=8$, $s_5=s_3+s_4=13$ и т. д. Полученная последовательность чисел

$$2, 3, 5, 8, 13, 21, 34, \ldots,$$

в которой каждый последующий член равен сумме двух предыдущих,— это хорошо известный в математике ряд Фибоначчи. О многих интересных свойствах чисел Фибоначчи и их разнообразных приложениях можно прочесть в популярной брошюре [21], а также в недавно изданной книге [6]. В частности, можно убедиться (см. [21]), что n-ый член ряда Фибоначчи вычисляется по формуле:

$$s_n = \frac{1}{\sqrt{5}}\left(\left(\frac{1+\sqrt{5}}{2}\right)^{n+2} - \left(\frac{1-\sqrt{5}}{2}\right)^{n+2}\right) \cong \frac{1}{\sqrt{5}}\,(1{,}62)^{n+2}.$$

3. Соединим оба предыдущих ограничения и найдем число двоичных слов постоянного веса t, не содержащих нескольких нулей подряд.

Рассуждать можно так. Пусть $q=n-t$ — число нулей в рассматриваемых словах. В любом слове имеется $q-1$ промежутков между ближайшими нулями, в каждом из которых находится одна или несколько

единиц (см. рис. 1). Предполагается, конечно, что $q \leqslant n/2$. В противном случае (при $q > n/2$) нет ни одного слова без рядом стоящих нулей.

Если из каждого промежутка удалить ровно по одной единице, то получим слово длины $n-q+1$, содержащее q нулей. Легко видеть, что любое такое слово может быть получено указанным образом из некоторого (и притом только одного) n-буквенного слова, содержащего q нулей, никакие два из которых не стоят рядом. Значит, искомое число совпадает с числом всех слов длины $n-q+1$, содержащих ровно q нулей, т. е. равно (см. дополнение 1)

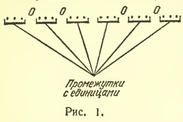


Рис. 1.

$$C_{n-q+1}^{q} = C_{t+1}^{n-t}.$$

4. Используя результаты дополнений 2, 3, убедиться в справедливости тождества:

$$\sum_{q=0}^{[n/2]} C_{n-q+1}^{q} = \frac{1}{\sqrt{5}}\left(\left(\frac{1+\sqrt{5}}{2}\right)^{n+2} - \left(\frac{1-\sqrt{5}}{2}\right)^{n+2}\right)$$

(символ $[n/2]$ означает наибольшее целое число, не превосходящее $n/2$).

5. При каком q число двоичных слов из дополнения 3 максимально?

6. Показать, что число всех n-буквенных d-ичных слов, в которых один из символов встречается фиксированное число t раз, равно $C_n^t (d-1)^{n-t}$ (ср. дополнение 1).

7. Обобщить результаты дополнений 2 и 3 применительно к d-ичному алфавиту.

2. ШИФРЫ, ШИФРЫ, ШИФРЫ...

Приемов тайнописи — великое множество, и, скорее всего, это та область, где уже нет нужды придумывать что-нибудь существенно новое. Наиболее простой тип криптограмм — это так называемые подстановочные криптограммы. Составляя их, каждой букве алфавита сопоставляют определенный символ (иногда тоже букву) и при кодировании всякую букву текста заменяют на соответствующий ей символ. В рассказе «Золотой жук» Эдгара По приводится как раз пример подстановочного шифра.

Автор рассказа наглядно демонстрирует, что расшифровка подобных криптограмм не составляет большой проблемы. Все основывается на том (за подробностями отсылаем читателя к оригиналу), что различные буквы естественного языка — английского, русского или какого-либо другого — встречаются в осмысленных текстах неодинаково часто. Следовательно, то же самое верно для соответствующих им знаков. В еще большей мере это относится к буквосочетаниям из двух или нескольких букв: лишь некоторые из них часты, многие же вообще не употребляются.

Анализируя частоту появления тех или иных знаков и их сочетаний (именно так поступает герой Эдгара По), можно с большой уверенностью восстановить буквы зашифрованного текста. Даже если в каких-то частях текста возникает неоднозначность, она легко устраняется по смыслу. Этот метод (он именуется частотным анализом) основывается, таким образом, на заранее известных частотах зашифрованных знаков. В следующей таблице указаны относительные частоты букв русского языка.

Буквы «е» и «ё», а также «ь», «ъ» кодируются обычно одинаково, поэтому в таблице они не различаются. Как явствует из таблицы, наиболее частая буква русского языка — «о». Ее относительная частота, равная 0,090, означает, что на 1000 букв русского текста приходится в среднем 90 букв «о». В таком же смысле понимаются относительные

Таблица 2

№	Буква	Относит. частота	№	Буква	Относит. частота	№	Буква	Относит. частота
0	а	0,062	10	к	0,028	20	ф	0,002
1	б	0,014	11	л	0,035	21	х	0,009
2	в	0,038	12	м	0,026	22	ц	0,004
3	г	0,013	13	н	0,053	23	ч	0,012
4	д	0,025	14	о	0,090	24	ш	0,006
5	е, ё	0,072	15	п	0,023	25	щ	0,003
6	ж	0,007	16	р	0,040	26	ы	0,016
7	з	0,016	17	с	0,045	27	ь, ъ	0,014
8	и	0,062	18	т	0,053	28	э	0,003
9	й	0,010	19	у	0,021	29	ю	0,006
						30	я	0,018

частоты и остальных букв. В таблице 2 не указан еще один «символ» — промежуток между словами. Его относительная частота наибольшая и равна 0,175.

С помощью таблицы 2 читатель сумеет, по-видимому, расшифровать такую криптограмму (расшифровку и пояснения см. в дополнении 1 на стр. 15):

Цярснсмщи ямякзж онкдждм мд снкыйн гкю онгрсямнб-нцмщф йпзоснвпялл мн б гптвзф рктцяюф нм ркнемдд.

Ненадежность подстановочных криптограмм (сравнительная легкость их расшифровки) была замечена уже дав-

но, и потому в разное время предлагались различные другие методы шифрования. Среди них важное место занимают перестановочные криптограммы. При их составлении весь текст разбивается на группы, состоящие из одинакового числа букв, и внутри каждой группы буквы некоторым образом переставляются. Если группа достаточно длинная (иногда это весь текст целиком), то число возможных перестановок очень велико, отсюда большое многообразие перестановочных криптограмм. Мы рассмотрим один тип перестановочной криптограммы, которая составляется при помощи так называемого ключевого слова. Буквы текста, который должен быть передан в зашифрованном виде, первоначально записываются в клетки прямоугольной таблицы, по ее строчкам. Буквы ключевого слова пишутся над столбцами и указывают порядок (нумерацию) этих столбцов способом, объясняемым ниже. Чтобы получить закодированный текст, надо выписывать буквы по столбцам с учетом их нумерации. Пусть текст таков: «В связи с создавшимся положением отодвигаем сроки возвращения домой. Рамзай». Используем для записи текста, в котором 65 букв, прямоугольную таблицу 11×6, в качестве ключевого возьмем слово из 6 букв «запись», столбцы занумеруем в соответствии с положением букв ключевого слова в алфавите. В результате получится следующая кодовая таблица:

Таблица 3

з	а	п	и	с	ь
2	1	4	3	5	6
в	с	в	я	з	и
с	с	о	з	д	а
в	ш	и	м	с	я
п	о	л	о	ж	е
н	и	е	м	о	т
о	д	в	и	г	а
е	м	с	р	о	к
и	в	о	з	в	р
а	щ	е	н	и	я
д	о	м	о	й	р
а	м	з	а	й	

Выписывая буквы из столбцов таблицы 3 (сначала из первого, затем из второго и т. д.), получаем такую шифровку:

Ссшоидмвщомвсвпноеиадаязмомирзноавоилевсоемзздсжоговийииаяетакряр

Ключевое слово известно, конечно, и адресату, который поэтому без труда расшифрует это сообщение. Но для тех, кто этим ключом не владеет, восстановление исходного текста весьма проблематично (хотя в принципе и возможно). Частотный анализ здесь по вполне понятным причинам не решает задачи. В лучшем случае, поскольку частоты букв примерно такие, как в таблице 2, он позволяет предположить, что было применено перестановочное кодирование.

Использование ключевого слова, конечно, не обязательно, можно было указать нумерацию столбцов цифровым ключом, в данном случае числом 214356. Слово удобнее, если ключ надо хранить в памяти (что немаловажно для конспирации).

Имеется ряд шифров, в которых совмещены приемы подстановочного и перестановочного кодирования. Шифр можно еще более усложнить, если дополнительно к этому каждую букву заменять не одним, а двумя или несколькими символами (буквами или числами). Вот пример. Расположим буквы русского алфавита в квадратной таблице 6×6 произвольным образом, например так, как в следующей таблице.

Таблица 4

	0	1	2	3	4	5
0		з	и	ы	р	с
1	а	т	у	й	ь	э
2		б	в	ф	к	л
3	м	ю	я	г	х	ц
4	ч	н	о		д	е
5	ж	ш	щ	п		

Каждую букву шифруем парой цифр: первая цифра это номер строки, в которой стоит данная буква, вторая — номер столбца. Например, букве «б» соответствует обозначение 21, а слову «шифр» — обозначение 51022304.

Еще бóльшие трудности для криптоанализа представляет шифр, связываемый с именем Тритемиуса. Этот шифр

является развитием рассматриваемого в дополнении 2 кода Цезаря и состоит в следующем. Буквы алфавита нумеруются по порядку числами 0, 1, ... , 30 (см. табл. 2). При шифровании ключевое слово (или номера его букв) подписывается под сообщением с повторениями, как показано ниже:

всвязиссоздавшимсяположениемотодвигаемсрокивозвра
записьзаписьзаписьзаписьзаписьзаписьзаписьзаписьз
щениядомойрамзай
аписьзаписьзапис

Каждая буква сообщения «сдвигается» вдоль алфавита по следующему правилу: буква с номером m, под которой стоит буква ключевого слова с номером k, заменяется на букву с номером $l=m+k$ (если $m+k<31$) или букву с номером $l=m+k-31$ (если $m+k\geqslant 31$). Например, первая буква «в» сдвигается на 7 букв и заменяется буквой «й», следующая буква «с» остается без изменения и т. д. Таким образом, номер l кодирующей буквы вычисляется по формуле:

$$l=m+k \pmod{31}. \qquad (1)$$

В цифровых обозначениях исходное сообщение и повторяемое ключевое слово запишутся в следующем виде:

Таблица 5

Сообщение	2	17	2	30	7	8	17	17	14	7	4	0	2
Ключ	7	0	15	8	17	27	7	0	15	8	17	27	7
Сообщение	24	8	12	17	30	15	14	11	14	6	5	13	8
Ключ	0	15	8	17	27	7	0	15	8	17	27	7	0
Сообщение	5	12	14	18	14	4	2	8	3	0	5	12	17
Ключ	15	8	17	27	7	0	15	8	17	27	7	0	15
Сообщение	16	14	10	8	2	14	7	2	16	0	25	5	13
Ключ	8	17	27	7	0	15	8	17	27	7	0	15	8
Сообщение	8	30	4	14	12	14	9	16	0	12	7	0	9
Ключ	17	27	7	0	15	8	17	27	7	0	15	8	17

После суммирования верхней и нижней строки по модулю 31 получаем последовательность чисел:

9.17.17.7.24.4.24.17.29.15.21.27.9.24.23.20.3.26.22.14.26, 22.23.1.20.8.20.20.0.14.21.4.17.16.20.27.12.12.1.24. 0. 6. 15.2.29.15.19.12.7.25.20.21.25.26.11. 14. 27. 22. 26. 12. 7. 12.22.8.26.

Наконец, заменяя числа на буквы, приходим к закодированному тексту:

йссзшдшсюпхьйшчфгыцоыцчбфиффаохдсрфьммбшажпвю пумзщфхщцылоьцымзмцины

Если ключевое слово известно, то дешифровка производится безо всякого труда на основе равенства:

$$m = l - k \pmod{31}.$$

Чрезвычайно трудно расшифровать подобный текст, если ключ неизвестен, хотя в истории криптографии были случаи, когда такие тексты разгадывались. Дело в том, что повторяемость ключевого слова накладывает некоторый отпечаток на криптограмму, а это может быть обнаружено статистическими методами, которые позволяют судить о длине ключевого слова, после чего расшифровка значительно упрощается.

Мы рассмотрели лишь некоторые способы составления криптограмм. Заметим, что комбинируя их, можно получать шифры, еще более труднодоступные для расшифровки. Однако вместе с этим возрастают трудности пользования шифром для отправителя секретного сообщения и адресата, поскольку сильно усложняется техника шифровки и дешифровки даже при наличии ключа.

Задачи и дополнения

1. Для расшифровки криптограммы иа стр. 11 подсчитаем, сколько раз встречается в ией каждая буква. Результаты подсчета приведеиы в следующей таблице:

Таблица 6

Буква	и	м	я	к	д	с	р	г	о	п	з	ф	ц	б	в	ж	й	л	т	щ	ю	е	и	ы
Число появлеиий в тексте	11	9	6	6	5	5	4	3	3	3	3	3	3	2	2	2	2	2	2	2	2	1	1	1

Наиболее часто встречающийся символ «и» скорее всего озиачает букву «о». Сделав такое предположеиие, рассмотрим следующий по частоте символ «м». В криптограмме имеется двубуквеиное сочетаиие «ми», и так как «и» — это «о», то символ «м» соответствует согласной.

Среди согласных в русском языке выделяются по частоте буквы «т» и «н» (см. табл. 2), и потому «м» скорее всего означает одну из этих букв. Разберем случай, когда «м» означает «н», предоставляя читателю самостоятельно убедиться, что другая возможность не приводит к осмысленной расшифровке криптограммы.

Если «м» — это «н», то в сочетании «мд», встречающемся в криптограмме, «д» означает скорее всего гласную. Из наиболее вероятных для «д» вариантов «а», «е», «и» выбираем «е», потому что лишь в этом случае имеющееся в криптограмме слово «ркнемдд» допускает осмысленную расшифровку. Итак три знака разгаданы: «н» — это «о», «м» — «н», «д» — «е». Обращаемся к сочетанию «ямякзж». В нем «я» может означать лишь гласную «а» или «и». Любые другие возможности заведомо не допускают разумного прочтения слова «ямякзж». Испытаем букву «а». Подставляя вместо «я» букву «а», вместо «м» — «н», вместо других знаков — точки, получим недописанное слово «ана...». В словаре имеется всего лишь несколько слов из 6 букв с таким началом: «анализ», «аналог», «ананас», «анатом». Из них годится лишь первое (почему?). Если вместо «я» подставить букву «и», то получится шестибуквенное сочетание с началом «ини», но в словаре нет ни одного такого слова. Расшифрованы еще четыре буквы: «я», «к», «з», «ж» означают соответственно «а», «л», «и», «з».

В слове «онкждм» известны все символы, кроме первого. Заменяя их буквами, получаем: «.олезен», Ясно, что неизвестная буква — это «п». Значит, «о» расшифровывается как «п».

Не разгаданы еще два сравнительно часто встречающиеся знака «о» и «р». Рассмотрим сочетание «ркнемдд», означающее «.ло.нее». Имеется немного вариантов его прочтения, один из них — «сложнее», и следовательно, скорее всего «р» — это «с», «е» — это «ж».

Из нерасшифрованных еще знаков чаще всего встречается «с». В соответствии с таблицей 2 среди оставшихся согласных наибольшую частоту имеет «т»,. Естественно поэтому предположить, что «с» означает «т».

Попытаемся восстановить зашифрованный текст, подставляя вместо разгаданных знаков соответствующие им буквы:

.астотн..анализ полезен не тол..о .л. по.стано.о.н.. ..ипто..а.. но . ....и. сл..а.. он сложнее

Ясны (по контексту), по крайней мере, три слова: «.астотн..» означает «частотный», «тол..о» — «только», «.л.» — «для». С учетом новой информации текст примет следующую форму:

Частотный анализ полезен не только для подстано.очны. к.ипто..а.. но . д...и. сл.чая. он сложнее

Окончательная расшифровка не представляет труда. Текст таков:

Частотный анализ полезен не только для подстановочных криптограмм, но в других случаях он сложнее

2. Шифр, примененный в предыдущем примере,— это так называемый шифр Цезаря. Он состоит в том, что весь алфавит сдвигается на определенное число букв вправо или влево. В данном случае был применен сдвиг влево на одну букву, т. е. каждая буква заменялась предшествующей буквой алфавита (при этом для буквы «а» предшествующей считалась буква «я»). Для шифра Цезаря имеется более простой способ расшифровки — так называемый метод полосок. На каждую полоску наносятся по порядку все буквы алфавита. В криптограмме

берется некоторое слово, например, «онкдждм». Полоски прикладываются друг к другу так, чтобы образовать данное слово (рис. 2). Двигаясь вдоль полосок, находим среди строк единственное осмысленное сочетание «полезен», которое и служит расшифровкой данного слова. Одновременно находим величину сдвига.

	Ж		Э	Я	Э	Е
И	З	Д	Ю	А	Ю	Ж
Й	И	Е	Я	Б	Я	З
К	Й	Ж	А	В	А	И
Л	К	З	Б	Г	Б	Й
М	Л	И	В	Д	В	К
Н	М	Й	Г	Е	Г	Л
О	Н	К	Д	Ж	Д	М
П	О	Л	Е	З	Е	Н
Р	П	М	Ж	И	Ж	О
С	Р	Н	З	Й	З	П
Т	С	О	И	К	И	Р
У	Т	П	Й	Л	Й	С
Ф	У	Р	К	М	К	Т
Х	Ф	С	Л	Н	Л	У
Ц	Х	Т	М	О	М	Ф

Рис. 2.

В качестве упражнения читателю рекомендуется расшифровать методом полосок следующую криптограмму, зашифрованную кодом Цезаря:

ЕИФИРРЛМ ФЕИХОЮМ ЗИРЯ НОСРОФВ Н ЕИЫИУЦ РСКСЕЮИ ХЦЫНЛ ФХСВОЛ ЕЮФСНС Е ВФОСП РИДИ Л НГКГССЯ РИ ТОЮОЛ ПЛПС Г ЦШСЗЛОЛ Е ФГПЦБ ЖОЦДЯ ОГКЦУЛ.

Ответом служит первая фраза романа И. С. Тургенева «Дворянское гнездо».

3. Расшифруйте числовую криптограмму:

77787935372412255432744327424864286438222386883852 2123

Ключ для расшифровки следующий. Разбейте числовую последовательность на двузначные числа. Вместо каждого числа надо подставить букву, стоящую на стр. 12 настоящей книги,— первая цифра числа указывает номер строки, в которой стоит искомая буква, а вторая — номер этой буквы в данной строке.

Это пример криптограммы, для составления и расшифровки которой используется некоторый заранее условленный текст, известный и отправнтелю, и адресату.

4. Другим примером шифра, использующего заранее условленный текст, является так называемый шифр «бегущего ключа». При шифровании по этому методу условленный текст накладывается на передаваемый так же, как в шифре Тритемиуса.

3. КОД ФАНО — ЭКОНОМНЫЙ КОД

Алфавита из двух (а подавно — из большего числа) символов, как мы убедились в § 1, достаточно для кодирования любого множества сообщений. Устанавливая этот факт, мы кодировали все сообщения словами одинаковой длины, что, однако, далеко не всегда бывает выгодно.

Представим себе, что одни сообщения приходится передавать довольно часто, другие — редко, третьи — совсем в исключительных случаях. Понятно, что первые лучше закодировать тогда короткими словами, оставив более длинные слова для кодирования сообщений, появляющихся реже. В результате кодовый текст станет в среднем короче, и на его передачу потребуется меньше времени.

Впервые эта простая идея была реализована упоминавшимся нами американским инженером Морзе в предложенном им коде. Рассказывают, что создавая свой код, Морзе отправился в ближайшую типографию и подсчитал число литер в наборных кассах. Буквам и знакам, для которых литер в этих кассах было припасено больше, он сопоставил более короткие кодовые обозначения (ведь эти буквы встречаются чаще). Так, например, в русском варианте азбуки Морзе буква «е» передается одной точкой, а редко встречающаяся буква «ц» — набором из четырех символов.

В математике мерой частоты появления того или иного события является его *вероятность*. Вероятность события A обозначают обычно символом $P(A)$ или просто буквой P. Не останавливаясь на определении вероятности, заметим только, что вероятность некоторого события (сообщения) можно представлять себе как долю тех случаев, в которых оно появляется, от общего числа появившихся событий (сообщений).

Так, если заданы четыре сообщения A_1, A_2, A_3, A_4 с вероятностями $P(A_1)=1/2$, $P(A_2)=1/4$, $P(A_3)=P(A_4)=1/8$, то это означает, что среди, например, 1 000 переданных сообщений около 500 раз появляется сообщение A_1, около 250 — сообщение A_2 и примерно по 125 раз — каждое из сообщений A_3 и A_4.

Эти сообщения нетрудно закодировать двоичными словами длины 2, например так, как показано в следующей таблице:

Таблица 7

A_1	A_2	A_3	A_4
00	01	10	11

Однако при таком кодировании вероятность появления сообщений никак не учитывается. Поступим теперь иначе. Разобьем сообщения на две равновероятные группы: в первую попадает сообщение A_1, во вторую — сообщения A_2, A_3, A_4. Сопоставим первой группе символ 0, второй — символ 1 (см. таблицу 8; во второй графе таблицы указаны вероятности сообщений).

Таблица 8

<table>
<tr><td>A_1</td><td>$1/2$</td><td>0</td><td></td><td></td></tr>
<tr><td>A_2</td><td>$1/4$</td><td rowspan="3">1</td><td>0</td><td></td></tr>
<tr><td>A_3</td><td>$1/8$</td><td rowspan="2">1</td><td>0</td></tr>
<tr><td>A_4</td><td>$1/8$</td><td>1</td></tr>
</table>

Это вполне в духе принципа, применявшегося в задаче с угадыванием. Действительно, символ 0 соответствует ответу «да» на вопрос «принадлежит ли сообщение первой группе?», а 1 — ответу «нет». Разница лишь в том, что раньше все множество разбивалось на две группы с одинаковым числом элементов, теперь же в первой группе один, а во второй — три элемента. Но, как и раньше, разбиение это таково, что оба ответа «да» и «нет» равновозможны. Продолжая в том же духе, разобьем множество сообщений A_2, A_3, A_4 снова на две равновероятные группы. Первой, состоящей из одного сообщения A_2, сопоставим символ 0, а второй, в которую входят сообщения A_3 и A_4,— символ 1. Наконец оставшуюся группу из двух сообщений разобьем на две группы, содержащие соответственно сообщения A_3 и A_4,

сопоставив первой из них 0, а второй — символ 1. Сообщение A_1 образовало «самостоятельную» группу на первом шаге, ему был сопоставлен символ 0, слово 0 и будем считать кодом этого сообщения. Сообщение A_2 образовало самостоятельную группу за два шага, на первом шаге ему сопоставлялся символ 1, на втором — 0; поэтому будем кодировать сообщение A_2 словом 10. Аналогично, для A_3 и A_4 выбираем соответственно коды 110 и 111. В итоге получается следующая кодовая таблица:

Таблица 9

A_1	A_2	A_3	A_4
0	10	110	111

Указанный здесь способ кодирования был предложен американским математиком Фано. Оценим тот выигрыш, который дает в нашем случае код Фано по сравнению с равномерным кодом, когда все сообщения кодируются словами длины 2. Представим себе, что нужно передать в общей сложности 1000 сообщений. При использовании равномерного кода на их передачу потребуется 2000 двоичных символов.

Пусть теперь используется код Фано. Вспомним, что из 1000 сообщений примерно 500 раз появляется сообщение A_1, которое кодируется всего одним символом (на это уйдет 500 символов), 250 раз — сообщение A_2, кодируемое двумя символами (еще 500 символов), примерно по 125 раз — сообщения A_3 и A_4 с кодами длины 3 (еще $3\times125+3\times125=$ $=750$ символов). Всего придется передать примерно 1750 символов. В итоге мы экономим восьмую часть того времени, которое требуется для передачи сообщений равномерным кодом. В других случаях экономия от применения кода Фано может оказаться еще значительнее.

Уже этот пример показывает, что показателем экономности или эффективности неравномерного кода являются не длины отдельных кодовых слов, а «средняя» их длина $\bar{l}$, определяемая равенством:

$$\bar{l} = \sum_{i=1}^{N} l_i P(A_i),$$

где l_i — длина кодового обозначения для сообщения A_i, $P(A_i)$ — вероятность сообщения A_i, N — общее число сообщений.

Наиболее экономным оказывается код с наименьшей средней длиной $\bar{l}$. В примере для кода Фано

$$\bar{l}=1\times 0{,}5+2\times 0{,}25+3\times 2\times 0{,}125=1{,}75,$$

в то время как для равномерного кода средняя длина $\bar{l}=2$ (она совпадает с общей длиной кодовых слов).

Нетрудно описать общую схему метода Фано. Располагаем N сообщений в порядке убывания их вероятностей: $P(A_1)\geqslant P(A_2)\geqslant\ldots\geqslant P(A_N)$. Далее разбиваем множество сообщений на две группы так, чтобы суммарные вероятности сообщений каждой из групп были как можно более близки друг к другу. Сообщениям из одной группы в качестве первого символа кодового слова приписывается символ 0, сообщениям из другой — символ 1. По тому же принципу каждая из полученных групп снова разбивается на две части, и это разбиение определяет значение второго символа кодового слова. Процедура продолжается до тех пор, пока все множество не будет разбито на отдельные сообщения. В результате каждому из сообщений будет сопоставлено кодовое слово из нулей и единиц.

Понятно, что чем более вероятно сообщение, тем быстрее оно образует «самостоятельную» группу и тем более коротким словом оно будет закодировано. Это обстоятельство и обеспечивает высокую экономность кода Фано.

Описанный метод кодирования можно применять и в случае произвольного алфавита из d символов с той лишь разницей, что на каждом шаге следует производить разбиение на d равновероятных групп.

Алгоритм кодирования Фано имеет очень простую графическую иллюстрацию в виде множества точек (вершин) на плоскости, соединенных отрезками (ребрами) по определенному правилу (такие фигуры в математике называют *графами*). Граф для кода Фано строится следующим образом (см. рис. 3). Из нижней (корневой) вершины графа исходят два ребра, одно из которых помечено символом 0, другое — символом 1. Эти два ребра соответствуют разбиению множества сообщений на две равновероятныег руппы, одной из которых сопоставляется символ 0, а другой — символ 1. Ребра, исходящие из вершин следующего «этажа», соответ-

Рис. 3.

ствуют разбиению получившихся групп снова на равновероятные подгруппы и т. д. Построение графа заканчивается, когда множество сообщений будет разбито на одноэлементные подмножества. Каждая концевая вершина графа, т. е. вершина, из которой уже не исходят ребра, соответствует некоторому кодовому слову. Чтобы указать это слово, надо пройти путь от корневой вершины до соответствующей концевой, выписывая в порядке следования по этому пути символы проходимых ребер. Например, вершине a_3 на рис. 3 соответствует слово 100, а вершине a_6 — слово 1110 (вершины, соответствующие кодовым словам, помечены на рисунке кружками).

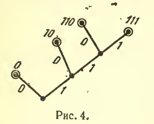


Рис. 4.

Граф для рассмотренного выше примера представлен на рис. 4.

Получающиеся для кодов Фано графы всегда обладают тем свойством, что они не содержат замкнутых контуров. Такие графы называют *деревьями* (мы будем называть их, учитывая происхождение, *кодовыми деревьями*). Кодовые деревья можно строить не только для кодов Фано, но и для других кодов. Независимо от алгоритма кодирования каждому дереву соответствует определенное множество кодовых слов. Например, для кодового дерева, изображенного на рис. 3, имеем:

$$a_1=00,\quad a_2=01,\quad a_3=100,\quad a_4=101,\quad a_5=110,$$
$$a_6=1110,\quad a_7=1111.$$

Кодовое дерево может быть построено для кода с произвольным основанием d. Каждое его ребро помечается тогда одним из d символов алфавита и из каждой вершины такого дерева исходит самое большее d различных ребер. Например, на рис. 5 представлено кодовое дерево для троичного кода со следующим множеством кодовых слов: 0, 10, 11, 120, 121, 20, 21, 220, 221, 222.

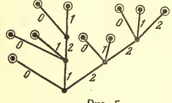


Рис. 5.

Кодовые деревья дают удобное геометрическое представление для многих важных понятий и облегчают, как мы увидим, решение различных задач, возникающих при построении экономных кодов.

Задачи и дополиеиия

1. Закодировать двончиым кодом Фано следующие мно- жсства сообщений:

а) семь сообщенний с вероятностямн

$$p_1=p_2=1/4; \quad p_3=p_4=p_5=1/8; \quad p_6=p_7=1/16;$$

б) десять сообщенний с вероятиостями

$$p_1=p_2=0{,}22; \quad p_3=p_4=p_5=p_6=0{,}1; \quad p_7=p_8=p_9=p_{10}=0{,}04.$$

Найти средиюю длииу каждого из полученных кодов.

Выяснить, каков вынгрыш по сравнеиию с равиомерным кодироваиием.

2. Прнведем прнмер тропчного коднроваиня методом Фаио для миожества нз 8 сообщений с вероятностямн

$$p_1=0{,}3; \quad p_2=p_3=p_4=0{,}15; \quad p_5=p_6=p_7=0{,}07; \quad p_8=0{,}04.$$

Таблнца 10

Сообщения	Вероятности				Кодовые слова
A_1	0,3	0			0
A_2	0,15	1	0		10
A_3	0,15		1		11
A_4	0,15		0		20
A_5	0,07		1	0	210
A_6	0,07	2		1	211
A_7	0,07		2	0	220
A_8	0,04			1	221

3. Закоднровать троичиым кодом Фано следующне множества сообщеиий:

а) 9 сообщеиий с вероятиостями

1/3; 1/9; 1/9; 1/9; 1/9; 1/9; 1/27; 1/27; 1/27;

б) 10 сообщенний с вероятиостями

0,2; 0,15; 0,1 5; 0,1; 0,1; 0,1; 0,05; 0,05; 0,05; 0,05.

4. СВОЙСТВО ПРЕФИКСА, ИЛИ КУДА ИДТИ РОБОТУ

При использовании неравномерных кодов приходится сталкиваться с одной проблемой, которую мы поясним на примере кодовой таблицы 9 предыдущего параграфа. На первый взгляд разумно было бы укоротить второе и третье кодовые слова, отбросив в них последний символ, так как при этом основное наше требование сохраняется: по-прежнему различные сообщения кодируются разными словами, как это видно из получившейся новой кодовой таблицы.

Таблица 11

A_1	A_2	A_3	A_4
0	1	11	111

Но пусть, к примеру, данные сообщения — это команды) выдаваемые электронному роботу: A_1 — идти прямо, A_2 — повернуть назад, A_3 — свернуть влево, A_4 — свернуть вправо. Предположим, что программа поведения робота задается следующей последовательностью:

$$A_1\ A_3\ A_1\ A_4\ A_1\ A_2\ A_3\ \ldots \qquad (1)$$

В результате кодирования эта последовательность преобразуется в такой двоичный текст:

01101110111 ... .

Легко вообразить себе, в какое недоумение привели бы мы робота, снабдив его подобной инструкцией. Куда же ему идти? Ясно, что сначала надо идти прямо. А дальше — свернуть влево или повернуть назад, потом еще раз назад? Впереди же еще бо́льшая путаница...

Естественно возразить, что следовало бы отделить одно кодовое слово от другого. Разумеется, это можно сделать, но лишь используя либо паузу между словами, либо специальный разделительный знак, для которого необходимо особое кодовое обозначение. И тот и другой путь приведет к значительному удлинению кодового текста, сводя на нет наше предыдущее «усовершенствование».

Другое дело, если мы будем пользоваться прежними кодовыми обозначениями для сообщений A_i. Тогда последовательность (1) будет закодирована так:

$$011001110101100\ \dots\ .$$

Здесь уже не может быть разночтений. Первое слово 0 — идти прямо, второе слово 110 — свернуть влево, так как в списке кодовых слов не значатся слова́ 1 и 11. В этом сплошном тексте одно за другим однозначно выделяются кодовые слова, и нет сомнений, что робот найдет правильную дорогу.

Итак, суть проблемы в том, что нужно уметь в любом кодовом тексте выделять отдельные кодовые слова без использования специальных разделительных знаков. Иначе говоря, мы хотим, чтобы код удовлетворял следующему требованию: всякая последовательность кодовых символов может быть единственным образом разбита на кодовые слова. Коды, для которых последнее требование выполнено, называются *однозначно декодируемыми* (иногда их называют *кодами без запятой*).

Наиболее простыми и употребимыми кодами без запятой являются так называемые *префиксные коды*, обладающие тем свойством, что никакое кодовое слово не является началом (префиксом) другого кодового слова. Если код префиксный, то, читая кодовую запись подряд от начала, мы всегда сможем разобраться, где кончается одно кодовое слово и начинается следующее. Если, например, в кодовой записи встретилось кодовое обозначение 110, то разночтений быть не может, так как в силу префиксности наш код не содержит кодовых обозначений 1, 11 или, скажем, 1101. Именно так обстояло дело для рассмотренного выше кода, который очевидно является префиксным.

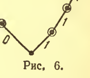


Рис. 6.

Нетрудно понять, как отражается свойство префиксности или его отсутствие на кодовом дереве. На рис. 6 представлено дерево для кода из таблицы 11 (кружками, как и раньше, помечены те вершины, которые соответствуют кодовым словам). Таким образом, если свойство префикса не выполняется, то некоторые промежуточные вершины дерева могут соответствовать кодовым словам. Для кода Фано это невозможно, так как по самому алгоритму кодирования построение кодового слова заканчивается одновременно с достижением концевой вершины. Следовательно, код Фано является префиксным кодом.

1. Префиксный код называют *полным*, если добавление к нему любого нового кодового слова (в данном алфавите) нарушает свойство префиксности. Убедиться, что двоичные коды, деревья которых изображены на рис. 3, 4, являются полными.

На рис. 7 представлено кодовое дерево префиксного, но неполного двоичного кода. Действительно, добавив к кодовым словам 0, 10, 111 слово 110, получим снова префиксный код.

2. Доказать, что двоичный код Фано является полным кодом. Верно ли аналогичное утверждение для кода Фано с произвольным основанием?

3. Проанализировав задачи 1 и 2, сформулировать необходимое и достаточное условие, которому должно удовлетворять кодовое дерево полного префиксного кода, и доказать его необходимость и достаточность.

4. Используя кодовое дерево, доказать, что всякий префиксный код может быть расширен до полного кода добавлением к нему некоторого множества кодовых слов.

5. Пусть k — максимальное значение длин кодовых слов префиксного кода. Показать, что число кодовых слов не превосходит величины 2^k в случае двоичного кода и величины d^k в случае кода с произвольным основанием d. При каких условиях достигается равенство?

6. Код, представленный на рис. 7, можно сделать более экономным, отбрасывая в слове 111 последний символ. При этом свойство префиксности не нарушится. Подобная операция, состоящая в том, что каждое



Рис. 7. Рис. 8.

слово префиксного кода заменяется наименьшим его началом, не являющимся началом других кодовых слов, называется *усечением*. Очевидно, что в результате усечения получается префиксный код и при этом более экономный, чем исходный. Возникают такие два вопроса:

1. Возможно ли усечение полного кода?

2. Можно ли утверждать, что в результате усечения получается полный код?

7. На рис. 8, *а* представлено кодовое дерево префиксного кода, для которого усечение невозможно. В то же время мы получим более экономный префиксный код (рис. 8, *б*), если в словах 110 и 111 вычеркнем второй символ 1.

Справедливо следующее утверждение: префиксный код является полным тогда и только тогда, когда его невозможно сократить (т. е. вычеркнуть хотя бы один знак в любом из кодовых слов), не нарушив свойства префикса.

8. Любопытно рассмотреть примеры однозначно декодируемых кодов, не обладающих свойством префикса. Пожалуй, простейшим примером такого рода является двоичный код {1, 10}. Ясно, что в любой кодовой последовательности, составленной из этих слов, всякое появле-

ние символа 1 означает начало нового кодового слова. Последнее остается справедливым для кода, каждое слово которого есть единица с последующими нулями. Разумеется, подобные коды далеко не самые экономные.

Приведем менее тривиальный пример однозначно декодируемого кода: {01, 10, 011}. Рекомендуем читателю указать алгоритм однозначного выделения кодовых слов из кодовой последовательности для этого кода.

9. Как узнать, является ли произвольный код однозначно декодируемым? Для этого можно предложить следующий способ. Возьмем всевозможные пары кодовых слов, в которых одно слово является префиксом другого. Для каждой такой пары найдем «повисший» суффикс, который остается после удаления префиксного слова из начальной части более длинного слова. Например, повисший суффикс для пары 10 и 10010 есть 010. Выпишем все повисшие суффиксы. Далее проделаем то же самое для каждой пары слов, состоящей из повисшего суффикса и кодового слова, в которой одно слово является префиксом другого. Выпишем все новые повисшие суффиксы, которые при этом получатся. Будем продолжать этот процесс до тех пор, пока будут появляться новые суффиксы. Код является однозначно декодируемым тогда и только тогда, когда никакой суффикс не совпадет ни с одним кодовым словом.

10. Выяснить, обладают ли свойством однозначной декодируемости следующие коды:

{110, 11, 100, 00, 10};
{100, 001, 101, 1101, 11011}.

11. Префиксные коды иногда называют *мгновенными* (или мгновенно декодируемыми), поскольку конец кодового слова опознается сразу, как только мы достигаем конечного символа слова при чтении кодовой последовательности. В этом состоит преимущество префиксных кодов перед другими однозначно декодируемыми кодами, для которых конец каждого кодового слова, как мы видели, может быть найден лишь после анализа одного или нескольких последующих символов, а иногда и всей кодовой последовательности. Таким образом, в отличие от префиксного кода декодирование здесь осуществляется с запаздыванием по отношению к передаче сообщения.

5. ЕЩЕ О СВОЙСТВЕ ПРЕФИКСА И ОДНОЗНАЧНОЙ ДЕКОДИРУЕМОСТИ

Возникает вопрос: каковы возможные длины кодовых слов однозначно декодируемого, в частности, префиксного кода? Понятно, например, что не существует двоичного префиксного кода с длинами кодовых слов 1, 1, 2. Несколько труднее ответить на такой вопрос: существует ли префиксный двоичный код, содержащий 100 слов с длинами от 1 до 100? Оказывается, существует. Кодовое дерево для требуемого кода, содержащее 100 «этажей», изображено на рис. 9 (пунктиром отмечены пропущенные этажи).

Вопросы такого рода можно было бы продолжить, отвечая на них в каждом конкретном случае. Но на самом деле можно установить общие условия (необходимые и достаточ-

ные) для существования префиксного и вообще произвольного однозначно декодируемого кода.

Пусть $V=\{a_1, a_2, \ldots, a_N\}$ — префиксный двоичный код, дерево которого схематически изображено на рис. 10.

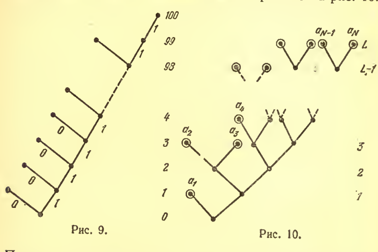


Рис. 9.

Рис. 10.

Пусть n_k — число кодовых слов длины k (n_k совпадает с числом концевых вершин k-го этажа). Конечно, справедливо неравенство

$$n_k \leqslant 2^k, \tag{1}$$

так как 2^k — максимально возможное число вершин на k-м этаже двоичного дерева. Однако в случае префиксного кода для n_k можно получить гораздо более точную оценку, чем (1). В самом деле, если $n_1, n_2, \ldots, n_{k-1}$ — число концевых вершин 1; 2; ... ; $k-1$ этажей дерева, то число всех вершин k-го этажа кодового дерева равно

$$2^k - 2^{k-1}n_1 - 2^{k-2}n_2 - \ldots - 2n_{k-1},$$

и потому

$$n_k \leqslant 2^k - 2^{k-1}n_1 - 2^{k-2}n_2 - \ldots - 2n_{k-1} \tag{2}$$

или иначе

$$2^{k-1}n_1 + 2^{k-2}n_2 + \ldots + 2n_{k-1} + n_k \leqslant 2^k.$$

Деля обе части последнего неравенства на 2^k, получаем:

$$\sum_{i=1}^{k} n_i 2^{-i} \leqslant 1. \tag{3}$$

Неравенство (3) верно для любого $k \leqslant L$ (L — максимальная длина кодовых слов), в частности

$$\sum_{i=1}^{L} n_i 2^{-i} \leqslant 1. \qquad (4)$$

Если $l_1, l_2, \ldots, l_N$ — длины кодовых слов $a_1, a_2, \ldots, a_N$, то неравенство (4) запишется в таком виде:

$$2^{-l_1} + 2^{-l_2} + \ldots + 2^{-l_N} \leqslant 1. \qquad (5)$$

Это и есть то условие, которому обязаны удовлетворять длины кодовых слов двоичного префиксного кода.

Оказывается, что неравенство (5), называемое в теории кодирования неравенством Крафта, является также достаточным условием того, чтобы существовал префиксный код с длинами кодовых слов $l_1, l_2, \ldots, l_N$.

Рассуждаем так. Если среди чисел $l_1, l_2, \ldots, l_N$ имеется ровно n_i чисел, равных i, то неравенство (5) можно переписать в виде (4), где L — максимальное из данных чисел. Из

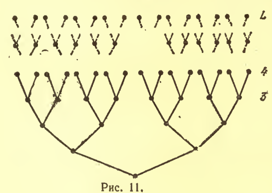


Рис. 11.

справедливости (4) подавно следует, что верны неравенства (3) для всех $k \leqslant L$, а, следовательно, и неравенство (2).

Обратимся к рис. 11, на котором изображено дерево «высоты» L, имеющее наибольшее число вершин и ветвей (ребер). Все концевые вершины (их 2^L) такого дерева находятся на последнем L-ом этаже, а из каждой вершины промежуточного этажа исходят ровно две ветви.

Для построения нужного префиксного кода мы должны подходящим образом выбрать n_1 слов длины 1, n_2 слов длины 2, вообще n_k слов длины k $(1 \leqslant k \leqslant L)$ или, иными словами, n_1 концевых вершин на первом, n_2 — на втором, ..., n_k — на k-ом этаже.

Из неравенства (2) при $k=1$ получаем $n_1 \leqslant 2$, т. е. требуемое число не превосходит общего числа вершин первого этажа. Значит, на этом этаже можно выбрать какие-то n_1 вершин в качестве концевых (n_1 равно 0, 1 или 2). Если это сделано, то из общего числа вершин второго этажа (их $2^2=4$) для построения кода можно использовать лишь $4-2n_1$ (почему?). Однако нам хватит и этого числа вершин, так как из неравенства (2) при $k=2$ вытекает

$$n_2 \leqslant 4-2n_1.$$

Аналогично, при $k=3$ имеем неравенство:

$$n_3 \leqslant 2^3-4n_1-2n_2.$$

Правая часть его вновь совпадает с допустимым для построения префиксного кода числом вершин третьего этажа, если на первых двух этажах уже выбраны n_1 и n_2 концевых вершин. Значит, снова можно выбрать n_3 концевых вершин на третьем этаже. Продолжая этот процесс вплоть до $k=L$, мы и получим требуемый код.

Если кодовый алфавит содержит d символов, то подобным же образом доказывается, что необходимым и достаточным условием для существования префиксного кода с длинами слов $l_1, l_2, \ldots, l_N$ является выполнение неравенства

$$d^{-l_1}+d^{-l_2}+\ldots+d^{-l_N} \leqslant 1. \qquad (6)$$

Оказывается, неравенству (6) обязаны удовлетворять и длины кодовых слов произвольного однозначно декодируемого кода. Поэтому, если существует однозначно декодируемый код с длинами слов $l_1, l_2, \ldots, l_N$, то существует и префиксный код с теми же длинами слов. Префиксными же кодами пользоваться удобнее по причине, указанной в дополнении **11** к § 4.

Задачи и дополнения

1. Каково минимальное число слов полного двоичного префиксного кода с максимальной длиной L? Какими будут длины кодовых слов такого минимального кода? (Ответы на эти вопросы подсказывает рис. 9.)

Решить те же вопросы в случае d-ичного кода.

2. Доказать, что префиксный код является полным тогда и только тогда, когда в неравенстве Крафта достигается равенство:

$$d^{-l_1}+d^{-l_2}+\ldots+d^{-l_N}=1.$$

Указание. То, что из предыдущего равенства вытекает полнота, очевидно. Для доказательства обратного утверждения следует предположить, что неравенство (6) строгое. Тогда из него выводится

строгое же неравенство

$$n_L < d^L - d^{L-1}n_1 - d^{L-2}n_2 - dn_{L-1},$$

которое показывает (почему?), что можно добавить по крайней мере еще одно слово, не нарушая префиксности.

3. Утверждение предыдущей задачи допускает следующую забавную интерпретацию, являющуюся одновременно и его доказательством (она заимствована из книги [6]).

Рассмотрим дерево, соответствующее полному префиксному коду. Представим себе, что на это дерево взбирается обезьяна. Начав с корня, она наугад выбирает любую из d исходящих из него ветвей; вероятность такого выбора равна $1/d$. Добравшись до очередной развилки, обезьяна снова наугад выбирает некоторую ветвь с вероятностью $1/d$ (напомним, что из каждой промежуточной вершины дерева полного кода исходит ровно d ветвей). Тогда вероятность того, что обезьяна достигнет какой-то определенной концевой вершины, находящейся на высоте k, равна $(1/d)^k$. Если таких вершин n_k, то с вероятностью $n_k d^{-k}$ обезьяна остановится на высоте k. На какой-то высоте от 1 до L обезьяне придется остановиться (вероятность этого равна 1). Поэтому $\sum_{k=1}^{L} n_k d^{-k} = 1$. (В приведенном рассуждении мы использовали правила сложения и умножения вероятностей.)

4. Префиксный код с данными длинами кодовых слов может быть построен далеко не единственным способом. Пусть d-ичный префиксный код (не обязательно полный) имеет n_k слов длины k $(1 \leq k \leq L)$. Доказать, что число различных таких кодов с фиксированными L и n_k равно произведению

$$\binom{d}{n_1} \times \binom{d^2 - dn_1}{n_2} \times \binom{d^3 - d^2 n_1 - dn_2}{n_3} \times \ldots \times \binom{d^L - d^{L-1}n_1 - \ldots - dn_{L-1}}{n_L},$$

где $\binom{i}{j} = C_i^j$ — число сочетаний из i элементов по j.

Например, число двоичных префиксных кодов с $L=4$, $n_1=0$, $n_2=1$, $n_3=2$, $n_4=4$ равно

$$C_2^0 C_4^1 C_6^2 C_8^4 = 4200.$$

5. Выше было доказано, что если для чисел $l_1, l_2, \ldots, l_N$ выполняется неравенство Крафта, то существует префиксный код с длинами $l_1, l_2, \ldots, l_N$. Найти этот код можно, строя этаж за этажом его кодовое дерево. Другой более удобный метод решения этой задачи был придуман Шенноном, и (применительно к двоичным кодам) он состоит в следующем.

Пусть числа $l_1, l_2, \ldots, l_N$ удовлетворяют неравенству

$$\sum_{i=1}^{N} 2^{-l_i} \leq 1.$$

Можно считать, что $l_1 \leq l_2 \leq \ldots \leq l_N$. Рассмотрим последовательность чисел

$$q_1 = 0;\; q_2 = 2^{-l_1};\; \ldots;\; q_j = \sum_{i=1}^{j-1} 2^{-l_i},\; \ldots,\; q_N = \sum_{i=1}^{N-1} 2^{-l_i}. \qquad (7)$$

Заметим, что все эти числа заключены в пределах $0 \leqslant q_j < 1$, поэтому каждое из них может быть представлено двоичной дробью вида $\sum_{k=1}^{n} \alpha_k 2^{-k}$, где каждое α_k есть 0 или 1. При этом из (7) можно заключить, что все эти дроби конечны, и двоичная запись для q_j имеет не более l_j значащих цифр. Таким образом, любое число q_j однозначно представимо в виде:

$$q_j = \sum_{i=1}^{l_j} c_{ij} 2^{-i},$$

где всякое c_{ij} есть 0 или 1. Следовательно, каждому q_j однозначно отвечает слово $v_j = c_{1j} c_{2j} \ldots c_{l_j j}$ длины l_j. Рассмотрим код $V = \{v_1, v_2, \ldots, v_N\}$. Покажем, что он обладает свойством префикса. Пусть v_j и v_k два слова ($k > j$). Тогда, согласно (7), $q_k - q_j \geqslant 2^{-l_j}$, а это означает, что l_j-й символ слова v_j не совпадает с l_j-м символом слова v_k. Следовательно, v_j не является началом v_k, откуда и вытекает префиксность кода V.

Разъясним сказанное на примере. Построим префиксный код с длинами слов $l_1 = 1$, $l_2 = l_3 = 3$, $l_4 = 4$. В этом случае

$$q_1 = 0; \quad q_2 = \frac{1}{2}; \quad q_3 = \frac{5}{8}; \quad q_4 = \frac{3}{4}.$$

Двоичная запись этих чисел с нужным числом l_j знаков следующая:

$$q_1 = 0, \quad q_2 = \frac{1}{2} + \frac{0}{2^2} + \frac{0}{2^3}; \quad q_3 = \frac{1}{2} + \frac{0}{2^2} + \frac{1}{2^3};$$

$$q_4 = \frac{1}{2} + \frac{1}{2^2} + \frac{0}{2^3} + \frac{0}{2^4}.$$

В результате мы получим искомый код:

$$v_1 = 0; \quad v_2 = 100; \quad v_3 = 101; \quad v_4 = 1100.$$

6. Используя метод Шеннона, найти префиксные коды с указанными ниже длинами слов:

а) $l_1 = l_2 = 2$; $\quad l_3 = l_4 = 3$; $\quad l_5 = l_6 = l_7 = 4$;

б) $l_1 = 1$; $\quad l_2 = 2$; $\quad l_3 = l_4 = l_5 = l_6 = 4$.

Построить кодовые деревья для полученных кодов. Определить, какой из этих кодов является полным.

6. ОПТИМАЛЬНЫЙ КОД

Как уже говорилось, общее правило при построении экономного кода следующее: чаще встречающиеся сообщения нужно кодировать более короткими кодовыми словами, а более длинные слова использовать для кодирования редких сообщений. Это правило и было реализовано в рассмотренном выше методе кодирования Фано. Но всегда ли метод Фано приводит к наиболее экономному коду? Ока-

зывается, нет. Способ построения оптимального кода, который мы здесь изложим, потребует от нас более тонких рассуждений.

Пусть сообщения $A_1, A_2, \ldots, A_N$ имеют вероятности $p_1, p_2, \ldots, p_N$ $(p_1 \geqslant p_2 \geqslant \ldots \geqslant p_N)$ и кодируются двоичными словами $a_1, a_2, \ldots, a_N$, имеющими длины $l_1, l_2, \ldots, l_N$. Постараемся выяснить, какими свойствами должен обладать двоичный код, если он оптимален.

1. В оптимальном коде менее вероятное сообщение не может кодироваться более коротким словом, т. е. если $p_i < p_j$, то $l_i \geqslant l_j$.

Действительно, в противном случае поменяем ролями кодовые обозначения для A_i и A_j. При этом средняя длина кодовых слов изменится на величину

$$p_i l_i + p_j l_j - p_i l_j - p_j l_i = (p_i - p_j)(l_i - l_j) > 0,$$

т. е. уменьшится, что противоречит определению оптимального кода.

2. Если код оптимален, то всегда можно так перенумеровать сообщения и соответствующие им кодовые слова, что $p_1 \geqslant p_2 \geqslant \ldots \geqslant p_N$ и при этом

$$l_1 \leqslant l_2 \leqslant \ldots \leqslant l_N. \qquad (1)$$

В самом деле, если $p_i > p_{i+1}$, то из свойства 1 следует, что $l_i \leqslant l_{i+1}$. Если же $p_i = p_{i+1}$, но $l_i > l_{i+1}$, то переставим сообщения A_i и A_{i+1} и соответствующие им кодовые слова. Повторяя эту процедуру нужное число раз, мы и получим требуемую нумерацию.

Из неравенств (1) следует, что сообщение A_N кодируется словом a_N наибольшей длины l_N.

3. В оптимальном двоичном коде всегда найдется, по крайней мере, два слова наибольшей длины, равной l_N, и таких, что они отличаются друг от друга лишь в последнем символе.

Действительно, если бы это было не так, то можно было бы просто откинуть последний символ кодового слова a_N, не нарушая свойства префиксности кода. При этом мы, очевидно, уменьшили бы среднюю длину кодового слова.

Пусть слово a_t имеет ту же длину, что и a_N и отличается от него лишь в последнем знаке. Согласно свойствам 1 и 2 можно считать, что $l_t = l_{t+1} = \ldots = l_N$. Если $t \neq N-1$, то можно поменять ролями кодовые обозначения a_t и a_{N-1}, не нарушая при этом неравенств (1).

Итак, всегда существует такой оптимальный код, в котором кодовые обозначения двух (наименее вероятных)

сообщений A_{N-1} и A_N отличаются лишь в последнем символе.

Отмеченное обстоятельство позволяет для решения задачи рассматривать только такие двоичные коды, у которых кодовые обозначения a_{N-1} и a_N для двух наименее вероятных сообщений A_{N-1} и A_N имеют наибольшую длину, отличаясь лишь в последнем символе. Это значит, что концевые вершины a_{N-1} и a_N кодового дерева искомого кода должны быть соединены с одной и той же вершиной a предыдущего «этажа» (см. рис. 12).

Рис. 12.

Рассмотрим новое множество сообщений $A^{(1)}=\{A_1, A_2, \ldots, A_{N-2}, A\}$ с вероятностями $p_1, p_2, \ldots, p_{N-2}, p=p_{N-1}+p_N$. Оно получается из множества $\{A_1, A_2, \ldots, A_{N-2}, A_{N-1}, A_N\}$ объединением двух наименее вероятных сообщений A_{N-1}, A_N в одно сообщение A. Будем говорить, что $A^{(1)}$ получается *сжатием* из $\{A_1, A_2, \ldots, A_{N-2}, A_{N-1}, A_N\}$.

Пусть для $A^{(1)}$ построена некоторая система кодовых обозначений $K^{(1)}=\{a_1, a_2, \ldots, a_{N-2}, a\}$, иными словами, указано некоторое кодовое дерево с концевыми вершинами $a_1, a_2, \ldots, a_{N-2}, a$. Этой системе можно сопоставить код $K=\{a_1, a_2, \ldots, a_{N-2}, a_{N-1}, a_N\}$ для исходного множества сообщений, в котором слова a_{N-1} и a_N получаются из слова a добавлением соответственно 0 и 1. Процедуру перехода от $K^{(1)}$ к K назовем *расщеплением*.

Справедливо следующее утверждение, открывающее путь для построения оптимального кода:

Рис. 13.

Если код $K^{(1)}$ для множества сообщений $A^{(1)}$ является оптимальным, то оптимален также и код K для исходного множества сообщений.

Для доказательства установим связь между средними длинами $\bar{l}$ и $\bar{l}'$ слов кодов K и $K^{(1)}$. Она, очевидно, такова:

$$\bar{l}=\bar{l}'+p. \tag{2}$$

Предположим, что код K не является оптимальным, т. е. существует код K_1 со средней длиной $\bar{l}_1<\bar{l}$. Как отмечалось, можно считать, что концевые вершины $\tilde{a}_{N-1}$ и $\tilde{a}_N$ его кодового дерева (см. рис. 13) соответствуют кодовым обозначениям для наименее вероятных сообщений A_{N-1} и A_N. Тогда эти обозначения отличаются лишь в последнем символе. Рассмотрим код $K_1^{(1)}=\{\tilde{a}_1, \ldots, \tilde{a}_{N-2}, \tilde{a}\}$, в котором слово $\tilde{a}$ получается из $\tilde{a}_{N-1}$ и $\tilde{a}_N$ отбрасыванием последнего сим-

вола. Средние длины $\bar{l}_1$ и $\bar{l}'_1$ связаны соотношением, аналогичным (2):

$$\bar{l}_1 = \bar{l}'_1 + p.$$

Из неравенства $\bar{l}_1 < \bar{l}$ следует $\bar{l}'_1 < \bar{l}'$, что противоречит оптимальности кода $K^{(1)}$. Утверждение доказано.

Теперь ясно, что для построения оптимального кода можно использовать последовательные сжатия исходного множества сообщений.

Проиллюстрируем процесс последовательных сжатий и расщеплений на примере множества из пяти сообщений с вероятностями $p_1=0{,}4$; $p_2=p_3=p_4=p_5=0{,}15$. Процесс этот отражен в следующей таблице:

Таблица 12

Сообщения	Вероятности и кодовые обозначения							
	Исходное множество		Сжатые множества					
			$A^{(1)}$		$A^{(2)}$		$A^{(3)}$	
A_1	0,4	1	0,4	1	0,4	1	0,6	0
A_2	0,15	010	0,3	00	0,3	00	0,4	1
A_3	0,15	011	0,15	010	0,3	01		
A_4	0,15	000	0,15	011				
A_5	0,15	001						

Каждое из множеств $A^{(1)}$, $A^{(2)}$, $A^{(3)}$ получается сжатием предыдущего множества. Множество $A^{(3)}$ состоит из двух сообщений, поэтому оптимальный код $K^{(3)}$ содержит два кодовых обозначения — 0 и 1. Последовательное расщепление $K^{(3)}$ дает оптимальный код для исходной системы сообщений.

Средняя длина $\bar{l}$ кодовых слов, равная $0{,}4+4\times3\times0{,}15=$ $=2{,}2$, является, как это следует из предыдущего, минимально возможной для данного множества сообщений.

Описанный метод кодирования был предложен в 1952 г. американским математиком Д. А. Хаффменом и называется его именем. Сравним теперь оптимальный код из таблицы 12 с кодом Фано для того же множества сообщений, который строится ниже.

Таблица 13

Сообщения	Вероятности				Кодовые слова
A_1	0,4	0	0		00
A_2	0,15		1		01
A_3	0,15	1	0		10
A_4	0,15		1	0	110
A_5	0,15			1	111

Подсчитаем среднюю длину $\bar{l}_F$ кодовых слов в этом случае:

$$\bar{l}_F = 2\times 0{,}4 + 2\times 2\times 0{,}15 + 2\times 3\times 0{,}15 = 2{,}3.$$

Следовательно, метод кодирования Фано не всегда приводит к оптимальному коду.

Как и метод Фано, метод кодирования Хаффмена может быть распространен на случай кодового алфавита, состоящего из произвольного числа символов. Этот вопрос рассмотрен в книге [2].

Задачи и дополнения

1. Доказать, что всякий оптимальный код является полным.

2. Закодировать двоичным кодом Хаффмена множество сообщений, имеющих вероятности:

$$p_1 = 0{,}25;\ p_2 = 0{,}2;\ p_3 = p_4 = p_5 = 0{,}15;\ p_6 = 0{,}1.$$

Построить соответствующее кодовое дерево.

3. Основываясь на алгоритме Хаффмена, найти способ непосредственного построения кодового дерева оптимального кода.

Указание. Построение дерева надо начинать не с корня, а с концевых вершин.

4. В некоторых случаях результаты двоичного кодирования по методу Фано те же, что и по методу Хаффмена, в том смысле, что длины соответствующих кодовых слов для обоих методов совпадают. Так, например, обстоит дело для множества сообщений с вероятностями:

$$p_1 = 1/4;\ p_2 = p_3 = p_4 = p_5 = 1/8;\ p_6 = p_7 = p_8 = p_9 = 1/16.$$

На самом деле справедливо следующее утверждение: если $p_i = 2^{-n_i}$ (т. е. если вероятности являются степенями двойки), то длины соответствующих кодовых слов в кодах Фано и Хаффмена одинаковы и равны n_i.

5. Не всегда полный код является оптимальным для данного множества сообщений. Можно доказать, однако, что всякий полный код

является оптимальным для некоторого множества сообщений с подходящим образом подобранными вероятностями. Так, если кодовые слова полного кода с основанием d имеют длины $n_1, n_2, \ldots, n_N$, то он оптимален для множества сообщений с вероятностями $d^{-n_1}, d^{-n_2}, \ldots, d^{-n_N}$.

7. ОБ ИЗБЫТОЧНОСТИ, ШУМАХ И КРИПТОГРАММЕ, КОТОРУЮ НЕЛЬЗЯ РАСШИФРОВАТЬ

Напомним читателю, что при кодировании сообщений нужно заботиться о том, чтобы их передача была достаточно быстрой, удобной и надежной. До сих пор мы интересовались лишь требованием быстроты. В этой связи были рассмотрены различные эффективные методы кодирования: коды Фано, Шеннона, Хаффмена. Последний, как было выяснено, является даже оптимальным при заданном множестве сообщений с заданными вероятностями. Указанные методы можно рассматривать как своего рода искусственные языки, предназначенные для экономной передачи информации. Оказывается, что привычные нам естественные языки (русский, английский и т. д.) являются в этом плане слишком расточительными и не выдерживают конкуренции с искусственными языками. Подтвердим это следующим мысленным экспериментом. Произвольный русский текст разобьем на куски одинаковой длины n (промежуток между словами можно по желанию либо игнорировать, либо считать отдельным символом). Получающиеся при этом различные буквосочетания из n букв будут встречаться, как это отмечалось прежде, не одинаково часто. Представим себе, что мы располагаем таблицей, в которой указаны всевозможные n-буквенные сочетания и их вероятности в русском тексте. Применяя метод Фано, закодируем их кодовыми словами, также использующими русский алфавит. Возвращаясь к исходному тексту, заменим каждый его кусок длины n соответствующим ему кодовым словом. Расчеты показывают, что действуя таким образом, мы смогли бы при достаточно большом n сократить исходный текст более, чем наполовину, сохранив при этом всю содержащуюся в нем информацию.

Было бы однако неосторожным на основе сказанного обвинить русский язык или другие естественные языки в несовершенстве. В теории информации имеется понятие, именуемое избыточностью языка или текста. Избыточность, точного определения которой мы не приводим, можно представлять себе как долю тех символов текста, которые

могут быть искажены или стерты без ущерба для его понимания, или иначе, как степень возможного «сжатия» текста в случае применения к нему методов оптимального кодирования. Мы вправе сказать поэтому, что естественные языки обладают высокой избыточностью (более 50%). Напротив, оптимально закодированные тексты имеют избыточность, близкую к нулю. Но именно высокая избыточность естественных языков позволяет с легкостью пользоваться ими в письменной и устной речи, например, свободно вникать в смысл написанного, несмотря на содержащиеся в тексте сокращения или опечатки, без особого труда понимать человека, говорящего с акцентом или на каком-нибудь диалекте и т. д. Герои известного романа Жюля Верна «Дети капитана Гранта» счастливой развязкой своих приключений также во многом обязаны избыточности языка.

Одним словом, избыточность позволяет языку противостоять влиянию всякого рода мешающих воздействий, в то время как оптимально закодированные тексты, оказывается, перед ними совершенно беззащитны. Подтвердим это приведенным в таблице 13 примером текста, закодированного двоичным кодом Фано.

Возьмем последовательность сообщений $A_2A_3A_5A_5A_1A_4$ и отвечающий ей кодовый текст 011011111100110. Пусть произошло искажение одного только первого символа. Получившаяся тогда кодовая последовательность 111011111100110 после расшифровки будет воспринята как последовательность сообщений $A_5A_2A_5A_4A_2A_3$.

Произошла непоправимая путаница, и ее виновником был всего лишь один неверно принятый кодовый символ. Аналогично обстоит дело с любым оптимальным кодовым текстом: ошибки (одна или несколько) переводят его в другой также вполне осмысленный кодовый текст, но смысл получившегося текста может быть совершенно отличен от первоначального. Такова расплата за оптимальность кода.

В то же время в реальных каналах связи, по которым происходит передача информации, ошибки неизбежны. Они являются следствием помех или, как иначе говорят, шумов, которые могут иметь самую различную физическую природу. Действие этих шумов на передаваемый текст можно ослабить, но устранить его полностью нельзя. Отсюда вывод: оптимальные коды в чистом виде непригодны для передачи сообщений по каналам связи с шумами, так как передача в этом случае становится ненадежной. С другой стороны, ясно, что надежность можно обеспечить только за счет некоторой избыточности кодового текста. В даль-

нейшем речь пойдет как раз о таких системах кодирования, которые предусматривают введение избыточности для борьбы с теми или иными видами ошибок. При построении подобных кодов всегда приходится идти на компромисс: с одной стороны, избыточность (т. е. количество дополнительных, «лишних», символов) не должна быть слишком велика, чтобы не растягивать время передачи; с другой стороны, помехоустойчивость (т. е. способность кода корректировать ошибки) должна быть достаточной, чтобы обеспечить надежность передачи.

Прежде чем переходить к рассмотрению помехоустойчивых кодов, позволим себе еще одно небольшое отступление в криптографию. К этому нас побуждает понятие избыточности, которое, оказывается, имеет прямое отношение к степени секретности криптограммы. Подстановочные криптограммы, например, потому-то и столь легки в расшифровке, что они, по существу, сохраняют избыточность первоначального текста.

Другие методы шифрования уменьшают эту избыточность, но все же не полностью ее устраняют, к тому же появляется избыточность, связанная со специфическими особенностями применяемого ключа. Совершенно секретная, т. е. недоступная для расшифровки, криптограмма должна быть освобождена как от избыточности исходного текста, так и от избыточности ключа. Способ составления такой криптограммы был предложен, как об этом уже упоминалось, К. Шенноном, и состоит он в следующем. Сначала устраняем избыточность текста, применяя к нему какой-нибудь из методов эффективного кодирования. Вслед за этим к получившемуся безызбыточному тексту применяем шифр со случайным ключом. Он похож на шифр Тритемиуса, для которого (применительно к русскому алфавиту)

$$l_i = m_i + k_i \pmod{31}.$$

В этом равенстве m_i и l_i по-прежнему являются номерами i-ой буквы шифруемого текста и криптограммы соответственно, а каждое k_i выбирается случайным образом среди чисел 0, 1, 2, ... , 30 — так что выбор любого из этих чисел в качестве k_i одинаково возможен.

Недостатком такой совершенно секретной системы является то, что вместе с шифрованным сообщением требуется посылать такое же по объему сообщение, содержащее информацию о случайном ключе, поскольку он заранее неизвестен адресату. Поэтому практически эта система малоприемлема.

Существуют, однако, системы шифрования, не использующие случайного ключа, и в то же время близкие к совершенно секретным. Необходимая система, например, может быть получена усовершенствованием шифра бегущего ключа (см. дополнение 4 к § 2). Она состоит в том, что на предварительно сжатый передаваемый текст накладывается другой текст, также предварительно сжатый.

Наконец, и от хлопотливой процедуры предварительного сжатия также можно отказаться, если в качестве ключа для шифровки использовать не один, а несколько (три или больше) условленных заранее текстов, прибавляя все их к передаваемому сообщению.

8. КОДЫ — АНТИПОДЫ

Мы выяснили, что при построении помехоустойчивых кодов не обойтись без дополнительных символов, которые должны быть присоединены к кодовым словам безызбыточного кода. Эти символы уже не несут информации о передаваемых сообщениях, но могут дать информацию о происшедших при передаче ошибках. Иными словами, их назначение — контролировать правильность передачи кодового слова. Вводимые дополнительные символы так и называют *контрольными* (или *проверочными*).

Самый незатейливый способ, позволяющий исправлять ошибки, состоит в том, что каждый информационный символ повторяется несколько раз, скажем, символ 0 заменяется блоком из n нулей, а символ 1 — блоком из n единиц. При декодировании n-буквенного блока, содержащего, быть может, ошибочные символы, решение принимается «большинством голосов». Если в принятом блоке нулей больше, чем единиц, то он декодируется как 00...0 (т. е. считается, что был послан нулевой символ), в противном случае — как 11...1. Такое правило декодирования позволяет верно восстановить посланные символы, если помехи в канале искажают менее половины символов в каждом передаваемом блоке. Если длину блока n выбрать достаточно большой, то мы практически обезопасим себя от возможных ошибок, однако передача сообщений будет идти черепашьими темпами. По этой причине указанный код (его называют кодом с повторением) не имеет большого практического значения, однако правило его декодирования («голосование») содержит в себе весьма полезную идею, которая с успехом применяется в других, практически более интересных помехоустойчивых кодах. Об этом речь пойдет дальше (см.

§§ 17, 18), а сейчас постараемся выяснить, на что мы можем рассчитывать при минимальной избыточности, когда к каждому кодовому слову добавляется всего лишь один проверочный символ. Пусть $\alpha_1\alpha_2...\alpha_n$ — двоичное кодовое слово. Выберем проверочный символ α_{n+1} с таким расчетом, чтобы на его значение одинаково влиял каждый из символов данного слова. Это естественное требование будет выполнено, если, например, положить $\alpha_{n+1}=\alpha_1+\alpha_2+...+\alpha_n$ (mod 2). Тогда проверочный символ α_{n+1} будет равен нулю, если в кодовом слове $\alpha_1\alpha_2...\alpha_n$ содержится четное число единиц, и единице — в противном случае. Например, присоединяя таким образом проверочный символ к слову 1010, получаем слово 10100, а из слова 1110 получим слово 11101.

Нетрудно видеть, что все удлиненные кодовые слова $\alpha_1\alpha_2...\alpha_n\alpha_{n+1}$ содержат четное число единиц, т. е.

$$\alpha_1+\alpha_2+\ldots+\alpha_n+\alpha_{n+1}=0 \pmod 2). \tag{1}$$

Допустим, что в процессе передачи в удлиненное кодовое слово $\alpha_1\alpha_2...\alpha_n\alpha_{n+1}$ вкралась одна ошибка (или даже любое нечетное число ошибок). Тогда в искаженном слове $\alpha_1'\alpha_2'...\alpha_n'\alpha_{n+1}'$ число единиц станет нечетным. Это и служит указанием на искажение в передаче слова. В конечном итоге все сводится к проверке соотношения (1) для символов принятого слова, что легко сделает простейшее вычислительное устройство — сумматор по модулю 2. Итак, правило приема следующее: если равенство (1) выполняется, считаем, что сообщение передано правильно, в противном случае отмечаем, что произошла ошибка и, когда это возможно, требуем повторить передачу кодового слова. Понятно, что иначе ошибки не исправить. Например, если принято «неправильное слово» 11100, то одинаково возможно, что было послано любое из кодовых слов:

01100, 10100, 11000, 11110, 11101.

Каждый из перечисленных случаев соответствует одной ошибке, а ведь ошибки могли произойти даже в трех, а то и в пяти символах.

Еще бóльшая неприятность подстерегает нас в случае двойной ошибки или вообще четного числа ошибок. Ведь тогда соотношение (1) не нарушится, и мы воспримем искаженное слово как верное.

Описанный код, который называют кодом с общей проверкой на четность, позволяет, следовательно, обнаружить

любое нечетное число ошибок, но «пропускает» искажения, если число ошибок четно.

Код с повторением и код с общей проверкой на четность — до некоторой степени антиподы. Возможности первого исправлять ошибки теоретически безграничны, но он крайне «медлителен». Второй очень быстр (всего один дополнительный символ), но зачастую «легкомыслен». В реальных каналах связи, как правило, приходится считаться с возможностью ошибок более чем в одном символе, поэтому в чистом виде код с общей проверкой на четность применяется крайне редко. Гораздо чаще применяют коды с несколькими проверочными символами (и, соответственно, с несколькими проверками на четность). Они позволяют не только обнаруживать, но и исправлять ошибки, и не только одиночные, но и кратные, и притом делать это гораздо эффективнее, чем упоминавшийся нами код с повторением. Это можно проиллюстрировать на одном красноречивом и в то же время простом примере.

Рассмотрим множество всех двоичных слов длины 9 (с их помощью можно закодировать $2^9=512$ сообщений). Расположим символы каждого слова $\alpha_1\alpha_2...\alpha_9$ в квадратной таблице следующим образом:

Таблица 14

α_1	α_2	α_3
α_4	α_5	α_6
α_7	α_8	α_9

К каждой строке и к каждому столбцу этой таблицы добавим еще по одному (проверочному) символу с таким расчетом, чтобы в строках и столбцах получившейся таблицы (таблица 15) было четное число единиц.

При этом, например, для первой строки и первого столбца будут выполняться проверочные соотношения:

$$\beta_1=\alpha_1+\alpha_2+\alpha_3 \pmod 2,$$
$$\beta_4=\alpha_1+\alpha_4+\alpha_7 \pmod 2$$

и аналогично для остальных строк и столбцов. Заметим, что

$$\beta_1+\beta_2+\beta_3=\beta_4+\beta_5+\beta_6 \pmod 2.$$

Обе эти суммы равны 0, если в слове $\alpha_1\alpha_2...\alpha_9$ четное число единиц, в противном случае обе они равны 1. Это дает воз-

Таблица 15

α_1	α_2	α_3	β_1
α_4	α_5	α_6	β_2
α_7	α_8	α_9	β_3
β_4	β_5	β_6	β_7

можность поместить в таблице 15 еще один проверочный символ β_7, равный

$$\beta_7=\beta_1+\beta_2+\beta_3=\beta_4+\beta_5+\beta_6 \pmod 2.$$

Например, слову 011010001 отвечает следующая таблица:

Таблица 16

0	1	1	0
0	1	0	1
0	0	1	1
0	0	0	0

Эти «маленькие хитрости» позволяют, оказывается, исправить любую одиночную ошибку, возникшую в процессе передачи, а сверх того и обнаружить любую двойную ошибку. В самом деле, если произошла одна ошибка, то нарушаются проверочные соотношения ровно для одной строки и ровно для одного столбца, как раз той строки и того столбца, на пересечении которых стоит ошибочный символ. Если же произошла двойная ошибка, то это приводит к нарушению проверок на четность либо в двух строках, либо в двух столбцах, либо сразу в двух строках и двух столбцах. По этим признакам мы и обнаруживаем двойную ошибку. (Однако исправить ее мы не можем — почему?)

Добавим к сказанному, что данный код позволяет обнаруживать многие, хотя и не все, ошибки более высокой

кратности (в трех, четырех и т. д. символах). Например, обнаруживаются тройные ошибки в символах α_1, α_2, α_3 или в символах α_1, α_2, α_6. А вот тройная ошибка в символах α_1, α_2, α_5 не может быть обнаружена, она будет воспринята как одиночная.

Продемонстрированное в этом примере сочетание проверок на четность по строкам и столбцам допускает широкие обобщения. О простейшем из них говорится в дополнении 3.

Задачи и дополнения

1. Для разобранного в данном параграфе примера найти число необнаруживаемых тройных ошибок. Какую часть они составляют от общего числа тройных ошибок?

2. Кодовые слова в примере на стр. 42 содержат 9 информационных символов и 7 проверочных, так что общая длина кодового слова равна 16. Такого числа символов достаточно, чтобы исправлять любые одиночные и обнаруживать любые двойные ошибки. Чтобы достичь того же эффекта для кода с повторением, нужно каждый информационный символ повторить 4 раза, так что общая длина кодового слова будет равна 36. Сравнение явно не в пользу кода с повторением. Справедливости ради надо все же отметить, что код с повторением обладает по сравнению с использованным в примере кодом некоторыми преимуществами в смысле обнаружения ошибок более высокой кратности. Предлагаем читателю найти, например, долю необнаруживаемых тройных ошибок для указанного кода с повторением и сравнить ответ с результатом задачи 1. Интересно найти также долю исправимых (!) тройных ошибок для этого кода.

3. Пример из данного параграфа обобщается следующим образом (смотри также дополнение 12 к § 11). Пусть каждое сообщение кодируется двоичным словом длины mn. Расположим все символы в прямоугольную таблицу (матрицу) с m строками и n столбцами:

Таблица 17

α_{11}	α_{12}		α_{1n}	$\alpha_{1,\,n+1}$
α_{21}	α_{22}		α_{2n}	$\alpha_{2,\,n+1}$
α_{m1}	α_{m2}		α_{mn}	$\alpha_{m,\,n+1}$
$\alpha_{m+1,\,1}$	$\alpha_{m+1,\,2}$		$\alpha_{m+1,\,n}$	$\alpha_{m+1,\,n+1}$

Как и прежде, добавим к каждой строке и к каждому столбцу по одному проверочному символу, так чтобы во всех строках и столбцах получились четные суммы. Аналогично прежнему выбираем символ $\alpha_{m+1,\ n+1}$. Полученное множество слов образует код, исправляющий любые одиночные и обнаруживающий любые двойные ошибки.

9. КОД ХЕММИНГА

Пусть количество сообщений, которые требуется передавать абоненту, равно 16. Для их безызбыточного кодирования можно использовать двоичные слова длины 4, но тогда код не будет корректировать ошибки. При использовании слов длины 5, как мы уже знаем, можно обнаружить, но не исправить любую одиночную ошибку. Впрочем, из дополнения 3 предыдущего параграфа вытекает, что если добавить 5 проверочных символов, то код сможет не только исправлять одиночные, но и обнаруживать двойные ошибки. Возникает вопрос: нельзя ли для этой цели обойтись меньшим количеством проверочных символов?

Вычислим сначала, каково минимальное число проверочных символов, необходимое для исправления любых одиночных ошибок. Нетрудно убедиться, что двух добавочных символов для этого недостаточно (предлагаем читателю проверить это самостоятельно).

Попробуем обойтись тремя проверочными символами, т. е. будем использовать для кодирования сообщений двоичные слова $\alpha_1\alpha_2\alpha_3\alpha_4\alpha_5\alpha_6\alpha_7$ длины 7. Наша задача — определить, произошла ли ошибка, и если произошла, то в каком месте. Но это то же самое, что указать одно из восьми чисел от 0 до 7 (0 соответствует отсутствию ошибки).

Пусть требуется передать сообщение, кодируемое словом $\alpha_1\alpha_2\alpha_3\alpha_4$. Добавим к этому слову три символа α_5, α_6, α_7, определяемые равенствами (здесь и до конца параграфа все равенства берутся по модулю 2):

$$\begin{aligned} \alpha_5 &= \alpha_2+\alpha_3+\alpha_4, \\ \alpha_6 &= \alpha_1+\alpha_3+\alpha_4, \\ \alpha_7 &= \alpha_1+\alpha_2+\alpha_4. \end{aligned} \qquad (1)$$

Если теперь нужно выяснить, допущена ли при передаче слова $\alpha_1\alpha_2\alpha_3\alpha_4\alpha_5\alpha_6\alpha_7$ одиночная ошибка в одном из символов α_4, α_5, α_6, α_7, то для этого достаточно вычислить сумму:

$$s_1=\alpha_4+\alpha_5+\alpha_6+\alpha_7. \qquad (2)$$

Ее значение, равное 1, соответствует ответу «да», значение 0 — ответу «нет» (почему?).

В случае «да» проверим, нет ли ошибки в символах α_6, α_7, в случае «нет» — не содержится ли ошибка в символах α_2, α_3. В каждом из этих случаев ответ дает значение суммы:

$$s_2=\alpha_2+\alpha_3+\alpha_6+\alpha_7. \qquad (3)$$

Если, например, значения обеих сумм (2) и (3) равны 1, то ошибка содержится либо в α_6, либо в α_7. Всего имеется четыре комбинации значений сумм s_1, s_2; они приведены в следующей таблице:

Таблица 18

s_1	s_2	Место ошибки
1	1	α_6 или α_7
1	0	α_4 или α_5
0	1	α_2 или α_3
0	0	нет ошибки или α_1

Наконец в каждом из четырех случаев нужно выбрать одну из двух возможностей. Это позволит сделать значение суммы

$$s_3=\alpha_1+\alpha_3+\alpha_5+\alpha_7. \qquad (4)$$

Итак, мы имеем три проверочных соотношения:

$$\begin{aligned} s_1&=\alpha_4+\alpha_5+\alpha_6+\alpha_7=0,\\ s_2&=\alpha_2+\alpha_3+\alpha_6+\alpha_7=0,\\ s_3&=\alpha_1+\alpha_3+\alpha_5+\alpha_7=0, \end{aligned} \qquad (5)$$

которые позволяют либо установить, что ошибки нет, либо однозначно указать ее место.

Отметим особо, что если произошла одиночная ошибка, то ее положение указывается числом с двоичной записью $s_1s_2s_3$. Пусть, например, $s_1=1$, $s_2=0$, $s_3=1$. Согласно таблице 18 ошибка допущена в четвертом или пятом разрядах; поскольку $s_3=1$, она — в пятом разряде, но $s_1s_2s_3=101$ как раз и есть двоичная запись числа 5.

Изученный здесь код — это код Хемминга длины 7 с четырьмя информационными символами.

В общем случае кодовые слова двоичного кода Хемминга, позволяющего исправить одиночную ошибку, имеют длину 2^m-1 (m — натуральное). Для определения положения ошибки тогда уже нужно m проверок, т. е. m прове-

рочных символов. Оставшиеся 2^m-1-m символов являются информационными. Проверки строятся по аналогии с рассмотренным случаем. Значения m проверок, как и выше, образуют номер положения ошибки.

Вернемся, однако, к вопросу, поставленному в начале этого параграфа. Добавим к кодовым словам кода Хемминга длины 7 еще один проверочный символ α_0, а к проверочным соотношениям (5) еще одно (общую проверку на четность):

$$s_0=\alpha_0+\alpha_1+\alpha_2+\alpha_3+\alpha_4+\alpha_5+\alpha_6+\alpha_7=0. \qquad (6)$$

Новый код по-прежнему будет содержать 16 кодовых слов, потому что, как и раньше, символы α_1, α_2, α_3, α_4 могут быть взяты какими угодно; по ним из соотношений (1) определяются символы α_5, α_6, α_7, а из равенства (6) и символ α_0. В случае одиночной ошибки добавленное соотношение (6) нарушается, а значения s_1, s_2, s_3 образуют номер положения ошибки. Если же произошла двойная ошибка, то соотношение (6) будет выполнено, а хотя бы одно из равенств (5) нарушится (почему?). Это и позволяет обнаружить любую двойную ошибку. Итак, для исправления одиночных и обнаружения двойных ошибок к четырем информационным символам достаточно добавить четыре проверочных символа. Можно показать, что обойтись меньшим числом проверочных символов невозможно.

Построенное множество кодовых слов $\alpha_0\alpha_1\alpha_2\alpha_3\alpha_4\alpha_5\alpha_6\alpha_7$, удовлетворяющих соотношениям (5) и (6),— пример расширенного кода Хемминга (длины 8 с четырьмя информационными символами).

Задачи и дополнения

1. Определить положение одиночной ошибки в искаженном слове 1100011 кода Хемминга длины 7.

2. Пусть 11010011 и 11001111— искаженные слова расширенного кода Хемминга длины 8. Какое из этих слов содержит одиночную, а какое — двойную ошибку? В случае одиночной ошибки определить ее положение.

3. К проверкам кода Хемминга длины 7 добавим (не меняя длины кода) общую проверку на четность:

$$\alpha_1+\alpha_2+\alpha_3+\alpha_4+\alpha_5+\alpha_6+\alpha_7=0. \qquad (7)$$

Сколько слов удовлетворяет соотношениям (5) и (7)?

4. Доказать, что код из задачи 3 исправляет одиночные и обнаруживает двойные ошибки.

5. Построить систему проверок для кода Хемминга длины 15. Сколько кодовых слов содержит этот код? Сколько информационных и сколько проверочных символов имеется в кодовом слове?

6. Еслн задан код Хеммннга длнны $n=2^m-1$, то существуют два простых способа получнть нз него коды с нсправленнем одннoffчных н обнаруженнем двойных ошнбок.

Первый нз ннх (он аналогнчен способу, разобранному в конце этого параграфа) таков: к кодовым словам добавляем проверочный снмвол α_0, а к проверочным соотношенням кода Хеммннга — общую проверку на четность: $\sum_{i=0}^{n} \alpha_i = 0$.

Во втором способе (см. задачу 3) длнна кодовых слов не меняется, но добавляется общая проверка на четность: $\sum_{i=1}^{n} \alpha_i = 0$.

Сколько ннформацнонных н проверочных снмволов содержнтся в каждом нз опнсанных здесь кодов?

10. НЕОБЫЧНОЕ ОБЫЧНОЕ РАССТОЯНИЕ

Известно, что расстояние между точками в пространстве определяется как длина отрезка прямой, соединяющей эти точки. Оно служит мерой близости точек — чем меньше расстояние, тем ближе друг к другу расположены точки. Если обозначать расстояние между точками a и b через $\rho(a, b)$, то для любых точек a, b и c имеем:

1) $\rho(a,b) \geqslant 0$;
2) $\rho(a,b)=0$ означает, что $a=b$;
3) $\rho(a,b)=\rho(b, a)$
4) $\rho(a,b)+\rho(b,c) \geqslant \rho(a,c)$.

(Последнее условие, называемое неравенством треугольника, означает, что длина любой стороны треугольника не превосходит суммы длин двух других сторон.)

Иногда удобно бывает определить меру близости или расстояние для элементов того или иного множества, даже если эти элементы и не являются точками в обычном смысле. При этом считается, что расстояние между элементами множества определено, если любым двум элементам a и b сопоставлено действительное число $\rho(a, b)$ и для любых элементов a, b и c данного множества выполняются условия 1) — 4).

Вернемся теперь к нашей основной теме. Мы уже знаем, что код тем лучше приспособлен к исправлению ошибок, чем больше отличаются друг от друга кодовые слова. Эту мысль можно будет выразить точно, если ввести расстояние на множестве n-буквенных слов.

Расстоянием $\rho(x, y)$ между двумя словами x и y назовем число несовпадающих позиций этих слов. Например, расстояние между словами $x=01101$ и $y=00111$ равно 2.

Определенное так расстояние называют *расстоянием Хемминга*. Не составляет большого труда проверить, что все свойства 1) — 4) расстояния в данном случае выполнены. Вопрос лишь в том, в какой мере это понятие поможет оценить способности кода исправлять и обнаруживать ошибки. Чтобы ответить на этот вопрос, введем для произвольного кода понятие *кодового расстояния*.

Кодовое расстояние $d(V)$ определим как минимальное расстояние между различными кодовыми словами из V:

$$d(V)=\min_{x \neq y} \rho(x, y).$$

Введенная только что величина является основным показателем корректирующих возможностей кода, поскольку верно следующее утверждение:

Код способен исправлять любые комбинации из t (и меньшего числа) ошибок тогда и только тогда, когда его кодовое расстояние больше $2t$.

В самом деле, если $d(V)>2t$, то для любых кодовых слов x, y имеем $\rho(x, y)\geqslant 2t+1$. Пусть при передаче некоторого слова x произошло $r\leqslant t$ ошибок, и в результате было принято слово z. Тогда $\rho(x, z)=r\leqslant t$, и в то же время расстояние $\rho(z, y)$ до любого другого кодового слова y больше t. Последнее вытекает из неравенства треугольника:

$$\rho(x, z)+\rho(z, y)\geqslant\rho(x, y)\geqslant 2t+1.$$

Значит, для восстановления посланного слова (декодирования) необходимо найти кодовое слово x, «ближайшее» к принятому слову z в смысле расстояния Хемминга. Подчеркнем, что это правило декодирования приводит к правильному результату, если число ошибок в передаваемом слове действительно не превосходило t.

Если же условие $d(V)>2t$ нарушается, то найдутся такие кодовые слова x и y, расстояние между которыми $\rho(x, y)\leqslant 2t$. Тогда может найтись такая комбинация из t ошибок в одном из слов x, что принятое слово z будет находиться от другого слова y не дальше, чем от x. Поэтому нельзя будет определить, какое из слов — x или y — было на самом деле передано

Задачи и дополнения

1. Имеется 8 двоичных слов длины 3. Их можно изобразить в пространственной системе координат как вершины куба со стороной 1. Каков в этом случае «геометрический смысл» расстояния Хемминга между словами?

2. Доказать, что для обнаружения s (или меньшего числа) ошибок необходимо и достаточно, чтобы кодовое расстояние удовлетворяло неравенству $d(V) \geqslant s+1$.

3. Доказать, что для исправления t (и меньшего числа) ошибок и вместе с этим обнаружения s (и меньших числа) ошибок ($s \geqslant t$) необходимо и достаточно, чтобы кодовое расстояние удовлетворяло неравенству $d(V) \geqslant t+s+1$.

4. Показать, что кодовое расстояние для кода с общей проверкой на четность равно двум, а для кода Хемминга — трем. Чему оно равно для кода с повторением, чему — для расширенного кода Хемминга?

11. ЛИНЕЙНЫЕ ИЛИ ГРУППОВЫЕ КОДЫ

Большинство рассмотренных выше кодов обладало следующим свойством: сумма (и разность) двух кодовых слов также являлась кодовым словом. Для кода с повторением это свойство очевидно. Ясно оно и для кода с общей проверкой на четность, потому что сумма двух слов с четным числом единиц есть также слово с четным числом единиц.

В § 9 был рассмотрен код Хемминга с системой проверочных соотношений (5). Обобщим теперь этот пример, выбрав в качестве кодового алфавита некоторое конечное поле F. Каждое слово $x_1x_2...x_n$ этого алфавита будем отождествлять с вектором $(x_1, x_2, \dots, x_n)$ n-мерного пространства L_n (в котором координаты векторов являются элементами F). Вектор, соответствующий кодовому слову, будем называть *кодовым*. Систему проверочных соотношений запишем в виде системы уравнений:

$$\begin{gathered} b_{11}x_1+b_{12}x_2+...+b_{1n}x_n=0, \\ b_{21}x_1+b_{22}x_2+...+b_{2n}x_n=0, \\ \dots\dots\dots\dots\dots\dots \\ b_{m1}x_1+b_{m2}x_2+...+b_{mn}x_n=0 \end{gathered} \qquad (1)$$

с коэффициентами b_{ij} из F. Код, состоящий из всех слов $x_1x_2...x_n$, для которых справедливы соотношения (1), называют *кодом с проверками на четность*.

Для такого кода выполняется следующее свойство: если векторы $(a_1, a_2, \dots, a_n)$ и $(b_1, b_2, \dots, b_n)$ являются кодовыми, а значит, решениями системы (1), то и их сумма $(a_1+b_1, a_2+b_2, \dots, a_n+b_n)$ также является решением этой системы и потому кодовым вектором. Справедливо и другое свойство решений системы (1): если α — элемент поля F и $(a_1, a_2, \dots, a_n)$ — решение системы (1), то и вектор $(\alpha a_1, \alpha a_2, \dots, \alpha a_n)$ также является решением системы (1).

Оба отмеченных свойства проверяются непосредственной подстановкой в систему (1) векторов

$$(a_1+b_1,\ a_2+b_2,\ \ldots,\ a_n+b_n) \text{ и } (\alpha a_1,\ \alpha a_2,\ \ldots,\ \alpha a_n).$$

Вместе эти свойства означают, что код с проверками на четность образует линейное подпространство в пространстве L_n всех n-буквенных слов. По этой причине коды с проверками на четность называют *линейными кодами* (двоичные линейные коды называют также *групповыми*). Если кодовое подпространство в пространстве L_n имеет размерность k, то употребляют для большей определенности термин линейный (n, k)-код.

Имеется очень много причин, по которым линейные коды являются важнейшими в теории кодирования. Одна из них связана с удобствами в обнаружении и исправлении ошибок, как это видно из примера, рассмотренного в § 9. Другая причина — это возможность компактного задания кода. Действительно, в случае линейного кода нет необходимости указывать полный список кодовых слов, ведь код вполне определен системой линейных уравнений (1) или матрицей этой системы *(проверочной матрицей)*:

$$H=\begin{pmatrix} b_{11} & b_{12} & \ldots & b_{1n} \\ b_{21} & b_{22} & \ldots & b_{2n} \\ \cdot & \cdot & \cdot & \cdot \\ b_{m1} & b_{m2} & \ldots & b_{mn} \end{pmatrix}.$$

В дальнейшем мы будем предполагать, что строки этой матрицы линейно независимы.

К числу других достоинств линейных кодов, которые связаны с предыдущими, относятся простые алгоритмы кодирования и декодирования, легко реализуемые электронными переключательными схемами. Вообще, можно сказать, что бурное развитие теории кодирования, которое происходило в последние десятилетия, объясняется главным образом тем, что к линейным кодам приложим хорошо развитый аппарат линейной алгебры и теории конечных полей.

Возвращаясь к рассмотренным ранее кодам, легко найдем их проверочные матрицы. Так, для кода с общей проверкой на четность имеем одно проверочное соотношение $x_1+x_2+\ldots+x_n=0$; соответственно этому проверочная матрица состоит из одной строки и имеет вид

$$H=(111\ldots1).$$

Из проверочных соотношений для кода с повторением

$$x_1 - x_2 = 0,$$
$$x_1 - x_3 = 0,$$
$$\cdots\cdots$$
$$x_1 - x_n = 0$$

получаем следующую проверочную матрицу:

$$H = \begin{pmatrix} 1 & -1 & 0 & \dots & 0 \\ 1 & 0 & -1 & \dots & 0 \\ \dots & \dots & \dots & \dots & \dots \\ 1 & 0 & 0 & \dots & -1 \end{pmatrix}.$$

Наконец, проверочная матрица двоичного кода Хемминга длины 7, как это следует из соотношений (5) § 9, выглядит так:

$$H = \begin{pmatrix} 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 \end{pmatrix}. \quad (2)$$

Имеется и другой способ матричного задания линейного кода. Он основан на том, что во всяком подпространстве линейного пространства можно выбрать базис, т. е. такую линейно независимую систему векторов, через которые линейно выражаются все вообще векторы подпространства. Пусть

$$\begin{aligned} a_1 &= (a_{11}, a_{12}, \dots, a_{1n}), \\ a_2 &= (a_{21}, a_{22}, \dots, a_{2n}), \\ &\cdots\cdots\cdots \\ a_k &= (a_{k1}, a_{k2}, \dots, a_{kn}) \end{aligned} \quad (3)$$

— базисные векторы линейного кода. Тогда всевозможные кодовые векторы исчерпываются линейными комбинациями

$$\alpha_1 a_1 + \alpha_2 a_2 + \dots + \alpha_k a_k, \quad (4)$$

где коэффициенты α_i — любые элементы исходного поля.

Таким образом, система базисных векторов (3) полностью определяет линейный (n, k)-код. Матрица G, составленная из них,

$$G = \begin{pmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ \dots & \dots & \dots & \dots \\ a_{k1} & a_{k2} & \dots & a_{kn} \end{pmatrix}, \quad (5)$$

называется *порождающей матрицей* кода.

Заметим, что базис можно выбрать не единственным образом, поэтому порождающая матрица G определена неоднозначно. Последнее, впрочем, верно и в отношении проверочной матрицы H.

Легко указать порождающую матрицу кода с повторением; она имеет вид:

$$G=(1\ 1\ 1 \ldots 1).$$

В качестве базисных векторов двоичного кода с общей проверкой на четность могут быть взяты, например, следующие $n-1$ векторов:

$$\begin{aligned} a_1 &= (1\ 1\ 0\ 0\ \ldots\ 0), \\ a_2 &= (1\ 0\ 1\ 0\ \ldots\ 0), \\ a_3 &= (1\ 0\ 0\ 1\ \ldots\ 0), \\ &\ldots\ldots\ldots\ldots \\ a_{n-1} &= (1\ 0\ 0\ 0\ \ldots\ 1). \end{aligned}$$

Поэтому матрица

$$G=\begin{pmatrix} 1\ 1\ 0\ 0 \ldots 0 \\ 1\ 0\ 1\ 0 \ldots 0 \\ 1\ 0\ 0\ 1 \ldots 0 \\ \ldots\ldots\ldots \\ 1\ 0\ 0\ 0 \ldots 1 \end{pmatrix}$$

является порождающей матрицей этого кода.

При использовании линейного кода для передачи сообщений полезно знать и порождающую, и проверочную матрицу. С помощью порождающей матрицы удобно кодировать сообщения. Поскольку линейный (n, k)-код с порождающей матрицей (5) имеет q^k кодовых слов (любой из k коэффициентов в (4) можно выбрать q способами), он позволяет закодировать все сообщения длины k в алфавите из q символов. Если $A=\alpha_1\alpha_2\ldots\alpha_k$ — такое сообщение (α_i — информационные символы), то самое удобное — сопоставить ему кодовый вектор a, совпадающий с линейной комбинацией (4) строк порождающей матрицы. Вектор a нетрудно записать в матричном виде, используя правило умножения матриц:

$$a=A\cdot G=(\alpha_1\alpha_2\ldots\alpha_k)\begin{pmatrix} a_{11} & a_{12} & a_{1n} \\ a_{21} & a_{22} & a_{2n} \\ \ldots & \ldots & \ldots \\ a_{k1} & a_{k2} & a_{kn} \end{pmatrix}. \qquad (6)$$

Пример. Пусть дана порождающая матрица двоичного линейного кода:

$$G=\begin{pmatrix} 1&1&1&0&0&0&0 \\ 1&0&0&1&1&0&0 \\ 0&1&0&1&0&1&0 \\ 1&1&0&1&0&0&1 \end{pmatrix}.$$

Этот код содержит $2^4=16$ кодовых слов, которыми можно закодировать все двоичные сообщения длины 4. Если, например, $A=(0101)$, то для соответствующего кодового вектора имеем:

$$a=(0\ 1\ 0\ 1)\begin{pmatrix} 1&1&1&0&0&0&0 \\ 1&0&0&1&1&0&0 \\ 0&1&0&1&0&1&0 \\ 1&1&0&1&0&0&1 \end{pmatrix}=(0\ 1\ 0\ 0\ 1\ 0\ 1).$$

О роли проверочной матрицы мы скажем в дальнейшем — она используется в основном при декодировании полученных сообщений и для исправления ошибок.

Выясним, как связаны порождающая и проверочная матрицы, и как по одной из них найти другую. Обратимся к соотношениям (1). Всякая строка порождающей матрицы, являясь кодовым вектором, удовлетворяет каждому из соотношений (1), т. е. для любых i и j

$$b_{i1}a_{j1}+b_{i2}a_{j2}+\ldots+b_{in}a_{jn}=0. \qquad (7)$$

Другими словами, любая строка порождающей и любая строка проверочной матриц ортогональны друг другу. Матричная запись равенств (7) выглядит так:

$$GH^T=0$$

(0 означает здесь нулевую матрицу).

Заметим также, что из соотношений (1) вытекает равенство

$$vH^T=0,$$

справедливое для каждого кодового вектора v.

Для отыскания порождающей матрицы нужно фактически найти $k=n-m$ линейно независимых решений системы (1). Эти решения как раз и будут строками порождающей матрицы.

Пример. Найдем порождающую матрицу кода Хемминга длины 7 с проверочной матрицей (2). В данном случае требуется найти 4 независимых решения следующей

системы:

$$\begin{aligned} x_4+x_5+x_6+x_7&=0,\\ x_2+x_3+\qquad\quad x_6+x_7&=0,\\ x_1\quad+x_3\quad+x_5\qquad+x_7&=0. \end{aligned}$$

Разрешаем систему относительно неизвестных x_1, x_2, x_4:

$$\begin{aligned} x_1&=x_3+x_5+x_7,\\ x_2&=x_3+x_6+x_7,\\ x_4&=x_5+x_6+x_7. \end{aligned} \tag{8}$$

Неизвестным x_3, x_5, x_6, x_7 можно придавать любые значения; тогда из равенств (8) могут быть определены оставшиеся неизвестные x_1, x_2, x_4. Придавая поочередно одному из неизвестных x_3, x_5, x_6, x_7 значение 1, а остальным — 0, получим 4 решения:

$$a_1=(1110000),\quad a_2=(1001100),\quad a_3=(0101010),$$
$$a_4=(1101001),$$

которые, как читатель может убедиться, линейно независимы. Составленная из этих решений матрица

$$G=\begin{pmatrix} 1&1&1&0&0&0&0\\ 1&0&0&1&1&0&0\\ 0&1&0&1&0&1&0\\ 1&1&0&1&0&0&1 \end{pmatrix}$$

и является искомой порождающей матрицей.

По поводу сказанного в этом параграфе возникает множество вопросов. Например, как, зная порождающую и проверочную матрицы, выяснить корректирующие способности данного кода; как реализовать эти способности (т. е. каким должен быть алгоритм декодирования, исправляющий или обнаруживающий ошибки); как, наконец, после исправления ошибок выделить информационные символы?

Остановимся вначале на последнем вопросе. Пусть, исправив ошибки, мы получили некоторый кодовый вектор $a=(a_1, a_2, \ldots, a_n)$. Тогда соответствующие ему информационные символы $\alpha_1, \alpha_2, \ldots, \alpha_k$ определяются из равенства (6). Иными словами, вектор $(\alpha_1, \alpha_2, \ldots, \alpha_k)$ есть решение (как можно показать,— единственное) системы линейных уравнений:

$$\begin{aligned} a_{11}\alpha_1+a_{21}\alpha_2+\ldots+a_{k1}\alpha_k&=a_1,\\ a_{12}\alpha_1+a_{22}\alpha_2+\ldots+a_{k2}\alpha_k&=a_2,\\ \ldots\ldots\ldots\ldots\ldots\ldots&\ldots\\ a_{1n}\alpha_1+a_{2n}\alpha_2+\ldots+a_{kn}\alpha_k&=a_n. \end{aligned}$$

Хотя задача отыскания этого решения и не содержит принципиальных трудностей, она может потребовать громоздких вычислений. Однако выделение информационных символов предельно упрощается, если использовать для кодирования так называемые систематические линейные коды.

Линейный (n, k)-код называется *систематическим*, если его порождающая матрица имеет вид

$$G=\begin{pmatrix} 1 & \ldots & 0 & \ldots & 0 & p_{11} & \ldots & p_{1m} \\ 0 & \ldots & 1 & \ldots & 0 & p_{21} & \ldots & p_{2m} \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ 0 & 0 & \ldots & 1 & & p_{k1} & \ldots & p_{km} \end{pmatrix} \qquad (9)$$

или вид

$$G=\begin{pmatrix} p_{11} & \ldots & p_{1m} & 1 & 0 & \ldots & 0 \\ p_{21} & \ldots & p_{2m} & 0 & 1 & \ldots & 0 \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ p_{k1} & \ldots & p_{km} & 0 & 0 & \ldots & 1 \end{pmatrix}, \qquad (9')$$

т. е. если она получается приписыванием к единичной матрице порядка k некоторой матрицы из k строк и $m=n-k$ столбцов (иначе говоря, матрицы порядка $k\times m$).

В случае (9) равенство (6) принимает вид:

$$a=(\alpha_1\alpha_2\ldots\alpha_k)\begin{pmatrix} 1 & 0 & \ldots & 0 & p_{11} & \ldots & p_{1m} \\ 0 & 1 & \ldots & 0 & p_{21} & \ldots & p_{2m} \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ 0 & 0 & \ldots & 1 & p_{k1} & \ldots & p_{km} \end{pmatrix}=$$

$$=(\alpha_1\alpha_2\ldots\alpha_k a_{k+1}\ldots a_n).$$

Таким образом, первые k символов любого кодового слова как раз и оказываются информационными, а остальные (они являются проверочными) связаны с информационными символами соотношениями:

$$\begin{aligned} a_{k+1}&=p_{11}\alpha_1+p_{21}\alpha_2+\ldots+p_{k1}\alpha_k, \\ a_{k+2}&=p_{12}\alpha_1+p_{22}\alpha_2+\ldots+p_{k2}\alpha_k, \\ &\ldots\ldots\ldots\ldots\ldots\ldots \\ a_n&=p_{1m}\alpha_1+p_{2m}\alpha_2+\ldots+p_{km}\alpha_k. \end{aligned} \qquad (10)$$

Равенства (10), очевидно, образуют систему проверочных соотношений систематического линейного кода. Очень важно, что всякий линейный код в некотором смысле эквивалентен систематическому (см. дополнение 9).

Дальнейшее прольет свет и на другие вопросы, поставленные выше.

Задачи и дополнения

1. *Весом* вектора u (обозначается $w(u)$) называют число его ненулевых координат. Понятно, что расстояние Хемминга между двумя векторами u_1 и u_2 равно весу их разности $w(u_1-u_2)$. Это позволяет упростить отыскание кодового расстояния линейного кода. Именно, справедливо следующее утверждение:

Кодовое расстояние линейного кода равно минимальному весу его ненулевых кодовых слов:

$$d(V) = \min_{\substack{v \in V \\ v \neq 0}} w(v).$$

Предоставляем читателю доказать это утверждение.

2. Пусть двоичный линейный код V содержит хотя бы одно слово нечетного веса. Доказать, что число всех таких слов составляет тогда ровно половину от общего числа кодовых слов.

Указание. Убедиться, что множество всех кодовых слов четного веса есть подпространство и найти смежные классы кода V по этому подпространству.

3. Рассмотрим матрицу порядка $q^k \times n$, в качестве строк которой взяты все кодовые векторы q-ичного линейного (n, k)-кода. Будем предполагать, что ни один столбец этой матрицы не является нулевым (иначе, вычеркнув нулевой столбец, мы получили бы код с тем же кодовым расстоянием, но меньшей длины). Показать, что в каждом столбце этой матрицы каждый из q элементов поля встречается ровно q^{k-1} раз. Пользуясь этим, убедиться, что суммарный вес всех кодовых слов равен $n(q-1)q^{k-1}$.

Указание. Проверить, что множество всех кодовых слов, содержащих 0 в некоторой фиксированной позиции, есть подпространство, и найти разложение кода в смежные классы по этому подпространству.

4. Предположим, что кодовый вектор $v = (x_1, x_2, \ldots, x_n)$ имеет вес, равный w, и $x_{k_1}, x_{k_2}, \ldots, x_{k_w}$ — его ненулевые координаты. Из проверочных соотношений (1) получаем тогда:

$$\begin{gathered}
b_{1k_1}x_{k_1} + b_{1k_2}x_{k_2} + \ldots + b_{1k_w}x_{k_w} = 0, \\
\cdots\cdots\cdots\cdots\cdots\cdots \\
b_{mk_1}x_{k_1} + b_{mk_2}x_{k_2} + \ldots + b_{mk_w}x_{k_w} = 0.
\end{gathered}$$

Это означает, что столбцы проверочной матрицы с номерами $k_1, k_2, \ldots, k_w$ линейно зависимы. Следовательно, каждому кодовому вектору веса w соответствует w линейно зависимых столбцов проверочной матрицы. Верно и обратное утверждение.

Отсюда вытекает, что кодовое расстояние линейного кода равно d тогда и только тогда, когда в проверочной матрице найдется d линейно зависимых столбцов, а всякая система из меньшего числа столбцов линейно независима.

5. Доказать, что двоичный линейный код исправляет любые одиночные ошибки тогда и только тогда, когда все столбцы его проверочной матрицы ненулевые и различные. Верно ли это для любого линейного кода?

6. Как изменится кодовое расстояние двоичного линейного кода при добавлении ко всем его словам одного проверочного символа, задающего общую проверку на четность?

7. Двоичный (8,4)-код задан порождающей матрицей

$$G=\begin{pmatrix} 1&1&1&0&1&0&0&0\\ 0&1&0&1&1&1&0&0\\ 0&0&1&1&0&1&0&1\\ 1&0&0&1&1&0&1&0 \end{pmatrix}. \qquad (11)$$

Найти его проверочную матрицу и кодовое расстояние.

То же задание для троичного (6,3)-кода с порождающей матрицей

$$G=\begin{pmatrix} 0&0&1&1&2&2\\ 1&1&1&2&0&0\\ 1&2&1&1&0&1 \end{pmatrix}.$$

8. Два кода V_1 и V_2 называются *эквивалентными*, если кодовые слова одного кода получаются из кодовых слов другого некоторой перестановкой символов (одной и той же для всех кодовых слов). Поскольку при этом веса кодовых слов остаются без изменений, то кодовые расстояния двух эквивалентных кодов совпадают, поэтому одинаковы также их возможности исправлять или обнаруживать ошибки. Ясно, что если порождающие матрицы кодов получаются одна из другой путем перестановки столбцов, то коды эквивалентны.

Эквивалентными являются, например, коды V_1 и V_2 со следующими порождающими матрицами:

$$G_1=\begin{pmatrix} 1&0&1&1&0&0\\ 0&0&1&0&1&1\\ 0&1&1&1&1&0 \end{pmatrix} \text{ и } G_2=\begin{pmatrix} 1&0&0&1&1&0\\ 0&1&0&1&0&1\\ 0&0&1&1&1&1 \end{pmatrix}.$$

Однако связь между матрицами эквивалентных кодов может быть и более сложной. Причина заключается в том, что порождающая матрица данного кода определена неоднозначно.

9. В предыдущем примере код V_1 эквивалентен систематическому коду V_2. Это не исключение, а правило, поскольку справедливо следующее утверждение:

Всякий линейный код эквивалентен систематическому коду.

Доказательство, которое рекомендуется продумать читателю, основывается, в сущности, на известном современному школьнику методе Гаусса исключения неизвестных.

Последовательность действий, приводящих произвольный код к систематическому, продемонстрируем на примере матрицы (11) (задача 7).

Вычитая из четвертой строки матрицы (11) первую, получим:

$$\begin{pmatrix} 1&1&1&0&1&0&0&0\\ 0&1&0&1&1&1&0&0\\ 0&0&1&1&0&1&0&1\\ 0&1&1&1&0&0&1&0 \end{pmatrix};$$

далее, вычитая из первой строки вторую, приходим к матрице

$$\begin{pmatrix} 1&0&1&1&0&0&0&0\\ 0&1&0&1&1&1&0&0\\ 0&0&1&1&0&1&0&1\\ 0&1&1&1&0&0&1&0 \end{pmatrix}.$$

Получающиеся при этих преобразованиях матрицы также порождают

исходный код. Осуществим теперь перестановку столбцов, чтобы слева получить единичную матрицу — пятый столбец поставим на место второго, восьмой — на место третьего, седьмой — на место четвертого столбца. В итоге получим порождающую матрицу систематического кода

$$\begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 1 & 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 \end{pmatrix}.$$

10. Показать, используя соотношения (7), что для систематического кода с порождающей матрицей (9) в качестве проверочной можно взять следующую матрицу:

$$H = \begin{pmatrix} -p_{11} & -p_{21} & \dots & -p_{k1} & 1 & 0 & \dots & 0 \\ -p_{12} & -p_{22} & \dots & -p_{k2} & 0 & 1 & \dots & 0 \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ -p_{1m} & -p_{2m} & \dots & -p_{km} & 0 & 0 & \dots & 1 \end{pmatrix}.$$

11. Пусть V — произвольный линейный (n, k)-код. Рассмотрим множество V' всех векторов пространства L_n, ортогональных каждому из кодовых векторов $v \in V$. Нетрудно проверить, что V' является подпространством в L_n и, следовательно, может рассматриваться как линейный код. Код V' называется *дуальным* к исходному коду V.

Убедиться, что если G и H — порождающая и проверочная матрицы кода V, то они служат соответственно проверочной и порождающей матрицей дуального кода V'.

Простейший пример кодов, дуальных друг к другу,— это код с повторением и код с общей проверкой на четность.

12. Одним из способов получения кодов, обладающих большим кодовым расстоянием, а значит, и высокой корректирующей способностью, является комбинирование двух или нескольких кодов. Примером такого комбинирования является рассматриваемая здесь конструкция. Она дает код, который является обобщением матричного кода из § 8 с проверками на четность по строкам и столбцам.

Пусть дано слово, содержащее $k=k_1k_2$ информационных символов. Разобьем множество этих символов на k_2 блоков по k_1 символов в каждом и запишем результат в виде квадратной матрицы порядка $k_2 \times k_1$:

$$\begin{pmatrix} \alpha_{11} & \alpha_{12} & \dots & \alpha_{1k_1} \\ \alpha_{21} & \alpha_{22} & \dots & \alpha_{2k_1} \\ \cdot & \cdot & \cdot & \cdot \\ \alpha_{k_21} & \alpha_{k_22} & \dots & \alpha_{k_2k_1} \end{pmatrix}. \qquad (12)$$

В первой строке этой матрицы выписаны по порядку символы первого блока, во второй — символы второго и т. д.

Рассмотрим теперь произвольный систематический линейный код V_1 длины n_1 с k_1 информационными символами и с тем же самым кодовым алфавитом. Строки матрицы (12) закодируем указанным кодом — для этого к каждой строке припишем n_1-k_1 проверочных символов таким образом, чтобы выполнялись проверочные соотношения кода V_1.

Далее каждый столбец получившейся матрицы порядка $k_2 \times n_1$,

$$\begin{pmatrix} \alpha_{11} & \alpha_{12} & \dots & \alpha_{1k_1} & \dots & \alpha_{1n_1} \\ \alpha_{21} & \alpha_{22} & \dots & \alpha_{2k_1} & \dots & \alpha_{2n_1} \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ \alpha_{k_21} & \alpha_{k_22} & \dots & \alpha_{k_2k_1} & \dots & \alpha_{k_2n_1} \end{pmatrix}$$

закодируем точно таким же способом с помощью линейного (n_2, k_2)-кода V_2.

В результате получим матрицу порядка $n_2 \times n_1$:

$$\begin{pmatrix} \alpha_{11} & \alpha_{12} & \dots & \alpha_{1k_1} & \dots & \alpha_{1n_1} \\ \alpha_{21} & \alpha_{22} & \dots & \alpha_{2k_1} & \dots & \alpha_{2n_1} \\ \dots & \dots & \dots & \dots & \dots & \dots \\ \alpha_{k_21} & \alpha_{k_22} & \dots & \alpha_{k_2k_1} & \dots & \alpha_{k_2n_1} \\ \dots & \dots & \dots & \dots & \dots & \dots \\ \alpha_{n_21} & \alpha_{n_22} & \dots & \alpha_{n_2k_1} & \dots & \alpha_{n_2n_1} \end{pmatrix}.$$

Выписывая элементы этой матрицы «по строкам», мы и составим кодовое слово, отвечающее исходному слову. Оно, очевидно, однозначно определяется матрицей (12).

Построенный код называется произведением кодов V_1 и V_2 и обозначается $V_1 \otimes V_2$.

Пусть теперь k, k_1, k_2— числа информационных символов кодов $V_1 \otimes V_2$, V_1 и V_2 а n, n_1, n_2— длины этих кодов. Из построения кода $V_1 \otimes V_2$ сразу же вытекает, что

$$k = k_1 k_2 \quad \text{и} \quad n = n_1 n_2.$$

Менее очевидно аналогичное соотношение между кодовыми расстояниями d, d_1, d_2 тех же кодов: $d = d_1 \cdot d_2$. Доказательство этого факта предоставляем читателю.

Предлагаем читателю также выяснить, какова связь между количеством ошибок, исправляемых кодами V_1, V_2 и $V_1 \otimes V_2$.

Проиллюстрируем сказанное примером. Пусть $k=8$, $k_1=4$, $k_2=2$. В качестве кода V_1 возьмем (7,4)-код Хемминга, записав предварительно его порождающую матрицу в систематической форме:

$$G = \begin{pmatrix} 1&0&0&0&1&1&0 \\ 0&1&0&0&1&0&1 \\ 0&0&1&0&0&1&1 \\ 0&0&0&1&1&1&1 \end{pmatrix},$$

в качестве V_2— (3,2)-код с общей проверкой на четность. Составим кодовое слово кода $V_1 \otimes V_2$, соответствующее исходному слову 00101110 с восемью информационными символами. Запишем его в виде матрицы порядка 2×4:

$$\begin{pmatrix} 0&0&1&0 \\ 1&1&1&0 \end{pmatrix}.$$

В результате кодирования строк (7,4)-кодом Хемминга (см. правило (6)) получим матрицу порядка 2×7:

$$\begin{pmatrix} 0&0&1&0&0&1&1 \\ 1&1&1&0&0&0&0 \end{pmatrix}.$$

Кодирование столбцов этой матрицы сводится к добавлению в каждом столбце символа общей проверки на четность. Получившейся в итоге матрице порядка 3×7

$$\begin{pmatrix} 0&0&1&0&0&1&1 \\ 1&1&1&0&0&0&0 \\ 1&1&0&0&0&1&1 \end{pmatrix}$$

соответствует следующее кодовое слово кода $V_1 \otimes V_2$:

001001111100001100011.

12. ДЕКОДИРОВАНИЕ ПО СИНДРОМУ И ЕЩЕ РАЗ О КОДЕ ХЕММИНГА

Слово «синдром» означает обычно совокупность признаков, характерных для того или иного явления. Такой же примерно смысл имеет понятие «синдром» и в теории кодирования. Синдром вектора, содержащего, быть может, ошибки, дает возможность распознать наиболее вероятный характер этих ошибок. Правда, определение, которое мы приводим ниже, не сразу позволяет это увидеть.

Синдромом вектора u называется вектор $s(u)$, определяемый равенством:

$$s(u) = uH^{\text{т}}.$$

Из правила перемножения матриц следует, что синдром есть вектор длины m, где m — число строк проверочной матрицы. В силу определения синдрома вектор u тогда и только тогда является кодовым ($u \in V$), когда его синдром равен нулевому вектору. В самом деле, равенство

$$uH^{\text{т}} = 0$$

равносильно тому, что координаты $x_1, x_2, \ldots, x_n$ вектора u удовлетворяют проверочным соотношениям (1) из § 11.

Пусть теперь вектор u не является кодовым, тогда этот вектор обязательно содержит ошибочные символы. Вектор u можно представить тогда в виде суммы посланного кодового вектора v (который пока не известен) и *вектора ошибки* e:

$$u = v + e. \qquad (1)$$

Ясно, что вектор $e = (\varepsilon_1, \varepsilon_2, \ldots, \varepsilon_n)$ содержит ненулевые символы в тех позициях, в которых вектор u содержит искаженные символы.

Важным обстоятельством является то, что синдромы принятого вектора u и вектора ошибки совпадают. Действительно,

$$s(u) = (v + e)H^{\text{т}} = vH^{\text{т}} + eH^{\text{т}} = eH^{\text{т}} = s(e). \qquad (2)$$

Рассмотрим теперь множество U всех векторов u', имеющих тот же синдром, что и вектор u. Пусть $a = u' - u$. Тогда

$$s(a) = (u' - u)H^{\text{т}} = u'H^{\text{т}} - uH^{\text{т}} = s(u') - s(u) = 0.$$

Так как $s(a) = 0$, то a — кодовый вектор. Обратно, если $u' - u$ — кодовый вектор, то $s(u') = s(u)$. Таким образом, для интересующего нас множества U имеем:

$$U = \{u + a \mid a \in V\}.$$

На языке теории групп это означает, что U есть смежный класс по подгруппе V (пространство L_n и его кодовое подпространство V можно рассматривать соответственно как группу и ее подгруппу относительно операции сложения векторов).

Сказанное позволяет сделать следующие выводы:

1. Два вектора имеют одинаковый синдром тогда и только тогда, когда они принадлежат одному смежному классу по кодовому подпространству. Таким образом, синдром вектора однозначно определяет тот смежный класс, которому этот вектор принадлежит.

2. Вектор ошибки $\boldsymbol{e}$ для вектора $\boldsymbol{u}$ нужно искать в силу равенства (2) в том же смежном классе, которому принадлежит и сам вектор $\boldsymbol{u}$.

Разумеется, указание смежного класса, которому принадлежит вектор $\boldsymbol{e}$, еще не определяет самого этого вектора. Естественно выбрать в качестве $\boldsymbol{e}$ тот вектор смежного класса, для которого вероятность совпадения с $\boldsymbol{e}$ наибольшая. Такой вектор называют *лидером* смежного класса. Если предположить, что большее число ошибок совершается с меньшей вероятностью, то в качестве лидера следует взять вектор наименьшего веса данного класса и в дальнейшем лидер будет пониматься именно в этом смысле.

При реализации алгоритма декодирования по синдрому составляют таблицу, в которой указываются синдромы s_i и лидеры $\boldsymbol{e}_i$ соответствующих им смежных классов. Алгоритм декодирования заключается тогда в следующем:

1. Вычисляем синдром $s(\boldsymbol{u})$ принятого вектора $\boldsymbol{u}$.

2. По синдрому $s(\boldsymbol{u})=\boldsymbol{s}_i$ определяем из таблицы лидер $\boldsymbol{e}_i$ соответствующего смежного класса.

3. Определяем посланный кодовый вектор $\boldsymbol{v}$ как разность

$$\boldsymbol{v}=\boldsymbol{u}-\boldsymbol{e}_i.$$

Может случиться, что в некоторых смежных классах окажется более одного лидера. Если искаженный вектор $\boldsymbol{u}$ попал в один из таких смежных классов, то разумнее отказаться от исправления ошибки, ограничившись ее обнаружением.

При всей своей простоте и прозрачности алгоритм синдромного декодирования обладает серьезным недостатком. Заключается он в том, что устройство, реализующее этот способ декодирования, должно хранить информацию о лидерах и синдромах. Объем же этой информации может оказаться очень большим даже при умеренных длинах кодовых слов (порядка нескольких десятков). В этом нетрудно

убедиться — ведь число лидеров и синдромов совпадает с числом смежных классов, которое по теореме Лагранжа равно $q^n : q^k = q^{n-k}$. Так что, например, для двоичного (50, 40)-кода получится 1024 лидеров и столько же синдромов, а для (50,30)-кода число их превзойдет миллион.

Таким образом, мы либо придем к чрезмерному усложнению аппаратуры для синдромного декодирования, либо практически вообще не сможем ее построить. К счастью, имеются коды, декодирование которых не требует обращения к таблице лидеров и синдромов. Простейшие из них — это код Хемминга и его расширенный вариант, рассмотренные в § 9.

Как мы уже знаем, двоичный код Хемминга является линейным, в общем случае имеет длину $n=2^m-1$, исправляет одиночные ошибки и обходится минимально возможным для этой цели числом проверок (это число равно m). Таким образом, проверочная матрица кода Хемминга имеет порядок $m \times (2^m-1)$. При этом все столбцы этой матрицы должны быть ненулевыми и различными (см. § 11, задача 5). Каждый столбец есть двоичный вектор длины m; всего имеется 2^m таких векторов, поэтому для построения проверочной матрицы кода Хемминга длины 2^m-1 нужно выписать (в качестве столбцов этой матрицы) все ненулевые двоичные векторы длины m. Порядок столбцов безразличен, но чаще всего их упорядочивают так, чтобы содержимое каждого столбца являлось двоичной записью его номера (сравни с матрицей (2) из § 11). Вот как выглядит проверочная матрица кода Хемминга длины 15 ($m=4$):

$$\begin{pmatrix} 0&0&0&0&0&0&0&1&1&1&1&1&1&1&1 \\ 0&0&0&1&1&1&1&0&0&0&0&1&1&1&1 \\ 0&1&1&0&0&1&1&0&0&1&1&0&0&1&1 \\ 1&0&1&0&1&0&1&0&1&0&1&0&1&0&1 \end{pmatrix}.$$

Алгоритм исправления одиночных ошибок в этом случае удивительно прост. Если вектор u содержит ошибочный символ в i-й позиции, то синдром $s(u)$ этого вектора совпадает с i-м столбцом проверочной матрицы. Таким образом, этот синдром, читаемый как двоичное число, и есть номер ошибочного символа.

Код Хемминга и в общем случае допускает усовершенствование того же рода, что и (7,4)-код из § 9. Добавление проверочного символа α_0, осуществляющего общую проверку на четность, приводит, как и там, к расширенному коду Хемминга с дополнительной способностью обнаруживать двойные ошибки. Его проверочная матрица легко может

быть получена из матрицы кода Хемминга: к каждой строке последней следует впереди приписать нулевой символ, а к получившимся строкам — строку из единиц, соответствующую общей проверке на четность:

$$\alpha_0+\alpha_1+\alpha_2+\ldots+\alpha_n=0.$$

Например, из приведенной выше проверочной матрицы для (15,11)-кода Хемминга получается следующая проверочная матрица для расширенного (16,11)-кода:

$$\begin{pmatrix} 1&1&1&1&1&1&1&1&1&1&1&1&1&1&1&1 \\ 0&0&0&0&0&0&0&0&1&1&1&1&1&1&1&1 \\ 0&0&0&0&1&1&1&1&0&0&0&0&1&1&1&1 \\ 0&0&1&1&0&0&1&1&0&0&1&1&0&0&1&1 \\ 0&1&0&1&0&1&0&1&0&1&0&1&0&1&0&1 \end{pmatrix}.$$

При вычислении синдрома искаженного вектора возможны две основные ситуации: либо синдром совпадает с одним из столбцов проверочной матрицы, либо это не так. Читатель легко проверит, что первая ситуация соответствует нечетному числу ошибок в слове, а вторая — четному.

В первом случае считаем, что произошла одиночная ошибка, и ее положение определяется номером столбца, с которым совпадает синдром.

Во втором случае считаем, что допущены две или любое большее четное число ошибок, если $s(u)\neq 0$. Если же $s(u)=0$, то, как обычно, полагаем, что ошибок при передаче не было.

Задачи и дополнения

1. Построить таблицу синдромов и соответствующих лидеров для (7,3)-кода с порождающей матрицей

$$G=\begin{pmatrix} 0&0&1&1&0&1&1 \\ 0&1&0&0&1&1&1 \\ 1&0&0&1&1&0&1 \end{pmatrix}.$$

2. Доказать, что алгоритм синдромного декодирования позволяет исправить любое количество ошибок, не превосходящее $\left[\frac{d-1}{2}\right]$, где d —кодовое расстояние.

Указание. Достаточно проверить, что все векторы веса $\left[\frac{d-1}{2}\right]$ и меньше попадают в различные смежные классы и, следовательно, являются лидерами в своих смежных классах.

3. Код с проверочной матрицей

$$H=\begin{pmatrix}1&0&1&0&0&0&0\\1&1&0&1&0&0&0\\0&1&0&0&1&0&0\\0&0&0&0&0&1&0\\0&0&0&0&0&0&1\end{pmatrix}$$

используется для исправления всех одиночных ошибок. Определить, какие еще ошибки могут быть в данном случае исправлены или обнаружены при декодировании по синдрому.

4. Проверить, что для обычного (нерасширенного) кода Хемминга лидеры ненулевых смежных классов исчерпываются всеми векторами веса 1. Верно ли это для расширенного кода Хемминга?

13. О КОДАХ, ИСПРАВЛЯЮЩИХ НЕСИММЕТРИЧНЫЕ ОШИБКИ

На практике нередко встречаются каналы, отличающиеся асимметричным характером ошибок, скажем, такие, в которых преобладают замещения вида $0 \to 1$ (т. е. вместо нуля принимается единица), а замещения $1 \to 0$ крайне редки.

Конечно, и в этом случае можно использовать коды, предназначенные для исправления замещений обоих видов, в частности, рассмотренный нами код Хемминга. Но это было бы слишком расточительно, так как корректирующая способность кода наполовину расходовалась бы тогда вхолостую. Придуманы поэтому коды, приспособленные специально к исправлению несимметричных ошибок.

Исходное соображение здесь очень простое: если в канале невозможны ошибки вида $1 \to 0$, то в принятом двоичном слове нет нужды проверять позиции с нулевыми символами — они наверняка переданы без искажений. Будем поэтому производить проверку таким образом, чтобы ее результат зависел только от позиций с единичными символами, точнее, от номеров этих позиций. С этой целью для произвольного двоичного слова $u=x_1x_2...x_n$ составим сумму

$$S(u)=\sum_{i=1}^{n} x_i i. \qquad (1)$$

В сумме (1) ненулевые слагаемые соответствуют только единичным символам и каждое из них совпадает с номером этого символа, т. е. число $S(u)$ равно сумме номеров единичных позиций слова u.

Как обычно, постараемся найти простое условие, выделяющее кодовые слова среди прочих. Будем искать это ус-

ловие в виде сравнения по некоторому модулю l. Представим себе, что число l уже выбрано, и рассмотрим код, состоящий из всех таких слов $v=x_1\ x_2\ \dots\ x_n$, для которых

$$S(v)\equiv 0 \pmod{l}, \qquad (2)$$

т. е. из слов, для которых сумма номеров единичных позиций делится на l без остатка. Обозначим указанный код через $V_{n,\,l}$. Так, при $n=4$, $l=5$ получим следующее множество кодовых слов:

$$V_{4,5}=\{0000,\ 1001,\ 0110,\ 1111\}.$$

Нетрудно убедиться, хотя бы перебором всех возможных случаев, что данный код исправляет любые одиночные ошибки вида $0\to 1$. Например, ошибка в третьем символе слова 1001 переводит его в слово 1011. При этом никакое другое кодовое слово не могло преобразоваться в результате одиночной ошибки в слово 1011. Поэтому получатель декодирует слово 1011 однозначно, считая, что было послано слово 1001. Аналогично обстоит дело с другими двоичными наборами, содержащими одиночные замещения нуля на единицу. Отметим, что в некоторых случаях (но не всегда!) возможно исправление также и двойных замещений нуля на единицу. Один из таких случаев — ошибка в двух первых символах слова 0000. Получающееся при этом слово 1100 не является кодовым и не могло получиться ни из какого кодового слова в результате одиночной ошибки. К тому же имеется единственный вариант двойного замещения, переводящего кодовое слово в слово 1100,— именно тот, который был указан выше.

Подобные свойства сохраняются и для произвольного кода $V_{n,l}$, если $l\geqslant n+1$. В частности, такой код исправляет любые одиночные ошибки вида $0\to 1$; при этом алгоритм их исправления чрезвычайно прост. Действительно, пусть в кодовом слове $v=x_1\ x_2\ldots x_n$ произошло не более одной ошибки и получилось слово u. Если ошибка произошла в j-м символе, то понятно, что $S(u)=S(v)+j$. Поскольку $S(v)\equiv 0 \pmod{l}$, то вычет числа $S(u)$ по модулю l равен j, т. е. номеру позиции, в которой произошла ошибка. Если же ошибки не произошло, то вычет $S(u)$ по модулю l равен нулю.

Проиллюстрируем исправление одиночной ошибки на примере рассмотренного выше кода $V_{4,5}$. Пусть принято слово $u=0111$. Тогда $S(u)=2+3+4=9\equiv 4 \pmod{5}$. Следовательно, ошибка произошла в четвертой позиции, т. е. передавалось кодовое слово 0110.

Наиболее употребимы коды $V_{n,l}$ при минимально возможном $l: l=n+1$. Именно они впервые были предложены советскими специалистами по кодированию Р. Р. Варшамовым и Г. М. Тененгольцем.

Коды $V_{n,l}$ обладают способностью исправлять еще один тип искажений кодовых слов, характерный для несимметричных каналов. Это так называемые выпадения и вставки символов.

Предположим, что в некотором слове $v=x_1x_2...x_n$ произошло выпадение одного символа, в результате чего получилось слово $u=y_1y_2...y_{n-1}$ длины $n-1$. Пусть n_1 — число единиц, а n_0 — число нулей, расположенных правее выпавшего символа. Оказывается, числа n_1 и n_0 могут быть определены с помощью суммы

$$S(u)=\sum_{i=1}^{n-1} iy_i.$$

В самом деле, если выпал символ 0, то $S(v)-S(u)=n_1$, а если выпал символ 1, то $S(v)-S(u)=n-n_0$. Если $w(u)$ — вес слова u, то, очевидно,

$$n_1 \leqslant w(u) < n-n_0 \leqslant n. \qquad (3)$$

Так как $S(v) \equiv 0 \pmod{l}$, то вычет числа $-S(u)$ по модулю l равен либо n_1 (в случае выпадения нуля), либо $n-n_0$ (в случае выпадения единицы). Неравенства (3) показывают, что если этот вычет не превосходит $w(u)$, то выпал символ 0, если же это не так, то символ 1. В первом случае выпавший символ 0 надо вставить в слово u так, чтобы правее него было число единиц, равное вычету числа $-S(u)$ по модулю l. Если же этот вычет больше, чем $w(u)$, то вставляем в слово u единицу так, чтобы правее нее было число нулей, равное разности n и вычета числа $-S(u)$. Если, например, при использовании кода $V_{4,5}$ принято слово $u=101$, то $S(u)=4$, а $w(u)=2$, вычет числа $-S(u)$ по модулю 5 равен 1, и, следовательно, выпал символ 0. Вставить его надо так, чтобы правее него была одна единица. Получается кодовое слово 1001.

Аналогично исследуется случай одиночной вставки символа. Читателю предлагается обосновать следующий алгоритм восстановления кодового слова, отвечающий этому случаю.

Пусть принято слово $u=y_1y_2...y_{n+1}$ длины $n+1$ и k — вычет числа $S(u)$ по модулю l. Тогда

1) если $k=0$, то отбрасывается последний символ слова u;

2) если $0 < k < w(u)$, то отбрасывается любой нулевой символ, правее которого в слове u ровно k единиц;

3) если $k = w(u)$, отбрасывается первый символ слова u;

4) если $k > w(u)$, отбрасывается любой единичный символ, правее которого имеется $n+1-k$ нулей.

Задачи и дополнения

1. Сравнить коды $V_{5,6}$ и $V_{5,10}$, а также коды $V_{6,7}$, $V_{6,8}, \ldots, V_{6,12}$ по их способности исправлять двойные замещения вида $0 \to 1$.

2. Для кода $V_{n,n+1}$ сформулировать признак исправимости замещения двух или большего числа символов. Каким будет этот признак для кода $V_{n,2n}$?

3. Найти алгоритм исправления (исправимого) двойного замещения для кодов $V_{n,n+1}$ и $V_{n,2n}$.

4. Коды $V_{n,k}$ можно использовать для исправления одиночных замещений вида $1 \to 0$. Каково в этом случае правило декодирования?

5. Построив код $V_{7,8}$, убедиться, что на местах с номерами 3, 5, 6, 7 встречаются всевозможные наборы из нулей и единиц и, значит, символы с этими номерами играют роль информационных.

6. Утверждение задачи 5 можно обобщить на случай любого кода $V_{n,n+1}$, для которого $n = 2^m - 1$.

Рассмотрим m позиций с номерами $1, 2, \ldots, 2^{m-1}$. Выберем произвольную комбинацию из нулей и единиц в оставшихся $n-m$ позициях. Тогда существует единственное заполнение позиций $1, 2, \ldots, 2^m$, для которого получившееся слово удовлетворяет условию (2), т. е. является кодовым. Отсюда вытекает, что число слов кода $V_{n,n+1}$ равно 2^{n-m}, т. е. совпадает с числом слов в коде Хемминга длины $n = 2^m - 1$.

7. Пусть $k \geqslant n+1$ есть простое число. Через $\tilde{V}_{n,k}$ обозначим код, состоящий из всех слов $v = x_1 x_2 \ldots x_n$, для которых выполняются два соотношения:

$$\sum_{i=1}^{n} i x_i \equiv 0 \pmod{k}, \quad \sum_{i=1}^{n} i^2 x_i \equiv 0 \pmod{k}.$$

Построить код $\tilde{V}_{6,7}$. Убедиться, что он исправляет любые одиночные и двойные замещения вида $0 \to 1$.

8. Показать, что всякий код $\tilde{V}_{n,k}$ (см. задачу 7) исправляет любые одиночные и двойные замещения вида $0 \to 1$. Сохранится ли это свойство для составного k?

14. ЦИКЛИЧЕСКИЕ КОДЫ

Среди линейных кодов особо важную роль играют так называемые *циклические коды*. По ряду причин они являются наиболее ценным достоянием теории кодирования. Во-первых, они допускают еще более компактное описание, чем произвольные линейные коды. Во-вторых, имеющиеся для линейных кодов алгоритмы кодирования и декодирования могут быть в применении к циклическим

кодам значительно упрощены; более того, для циклических кодов существуют свои особые методы декодирования, неприложимые к другим линейным кодам. Наконец, по своей структуре эти коды идеально приспособлены к реализации в современных технических устройствах.

Простые в реализации, они, однако, не столь просты в теории. Для полного разъяснения большинства вопросов, связанных с построением и методами декодирования циклических кодов, требуется довольно сложный алгебраический аппарат. Поэтому наше предстоящее знакомство с циклическими кодами будет далеко не полным.

Начнем с понятия циклического сдвига вектора. Пусть задан произвольный n-мерный вектор $a=(a_0, a_1, a_2, \dots, a_{n-1})$ с координатами из любого поля F (нумерацию координат в случае циклических кодов удобно начинать с нуля). *Циклическим сдвигом* этого вектора назовем вектор $a'=(a_{n-1}, a_0, a_1, a_2, \dots, a_{n-2})$. Например, для вектора (01101) последовательными циклическими сдвигами являются такие вектора:

$$(10110), \quad (01011), \quad (10101), \quad (11010).$$

Циклическим кодом называется линейный код, который вместе с любым своим вектором содержит также и его циклический сдвиг. Иными словами, циклический код обладает следующим свойством: циклический сдвиг любого кодового вектора снова является кодовым вектором.

Циклическим является, например, код с порождающей матрицей

$$G=\begin{pmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \end{pmatrix}.$$

Менее тривиальным примером циклического кода (проверка предоставляется читателю) является код, эквивалентный (7,4)-коду Хемминга, с проверочной матрицей

$$H=\begin{pmatrix} 1 & 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 1 & 1 & 0 & 1 \end{pmatrix}. \tag{1}$$

При рассмотрении циклических кодов кроме обычных действий над векторами мы имеем дело фактически еще с одной операцией, сопоставляющей каждому вектору его циклический сдвиг. Удобным алгебраическим средством для ее описания являются многочлены. С каждым вектором $a=(a_0, a_1, \dots, a_{n-1})$ свяжем многочлен $a(x)=a_0+a_1x+\dots+a_{n-1}x^{n-1}$, коэффициенты которого совпадают с соответ-

ствующими координатами вектора. В дальнейшем будем отождествлять вектор $\boldsymbol{a}$ с соответствующим ему многочленом $a(x)$, свободно переходя от векторной записи к записи в виде многочлена. Циклическому сдвигу $\boldsymbol{a}'=(a_{n-1}, a_0, a_1, \ldots, a_{n-2})$ вектора $\boldsymbol{a}$ соответствует тогда многочлен $a'(x)=a_{n-1}+a_0x+a_1x^2+\ldots+a_{n-2}x^{n-1}$. Сравним многочлены $a(x)$ и $a'(x)$. Если сумму первых $n-1$ слагаемых $a(x)$ умножить на x, мы получим сумму последних $n-1$ слагаемых многочлена $a'(x)$. Вообще, нетрудно заметить, что

$$a'(x)=x\,a(x)-a_{n-1}(x^n-1). \qquad (2)$$

Будем теперь считать, что x — образующий элемент циклической группы порядка n. Тогда n-я степень x равна единице:

$$x^n=1, \qquad (3)$$

а все меньшие степени, $1, x, x^2, \ldots, x^{n-1}$, являются различными элементами. При этом правило умножения степеней x следующее:

$$x^kx^m=x^r, \qquad (4)$$

где $r\equiv k+m \pmod n$ и $0\leqslant r<n$.

С учетом (3) равенство (2) дает:

$$a'(x)=xa(x),$$

т. е. циклический сдвиг любого вектора получается умножением этого вектора на x.

Из равенств (3) и (4) вытекает следующее правило умножения любых двух многочленов от x степени $\leqslant n-1$. Если

$$a(x)=a_0+a_1x+\ldots+a_{n-1}x^{n-1}$$

и

$$b(x)=b_0+b_1x+\ldots+b_{n-1}x^{n-1}$$

— два таких многочлена, то чтобы найти их произведение, сначала обычным образом раскрываем скобки, умножая степени x^k и x^m по правилу (4), а коэффициенты a_k и b_m — по правилу умножения в поле F, после чего приводим подобные члены.

Пример 1. Пусть $n=5$ и $a(x)=1+x+x^2+x^4$ и $b(x)=1+x^3+x^4$ — двоичные многочлены. Тогда

$$a(x)b(x)=(1+x+x^2+x^4)(1+x^3+x^4)=$$
$$=1+x+x^2+x^4+x^3+xx^3+x^2x^3+x^4x^3+x^4+xx^4+x^2x^4+x^4x^4=$$
$$=1+x+x^2+x^4+x^3+x^4+1+x^2+x^4+1+x+x^3=1+x^4.$$

Пример 2. Пусть $n=4$, $a(x)=1+2x+x^2$ и $b(x)=2+x+x^3$ — многочлены над полем Z_3. Имеем:

$$a(x)b(x)=(1+2x+x^2)(2+x+x^3)=$$
$$=2+2\cdot 2x+2x^2+x+2x^2+x^3+x^3+2xx^3+x^2x^3=$$
$$=2+x+2x^2+x+2x^2+2x^3+2+x=1+x^2+2x^3.$$

Итак, для многочленов кроме операции сложения определена также и операция умножения (напомним, что сложение многочленов не нуждалось в специальном определении, поскольку многочлены понимаются нами как векторы). При этом, очевидно, операция умножения многочленов коммутативна, ассоциативна и дистрибутивна относительно операции сложения. Поэтому множество F_n всех многочленов степени $\leqslant n$ образует относительно указанных операций кольцо. Но тогда циклический код может быть описан в чисто алгебраических терминах следующим образом:

Линейный код V является циклическим тогда и только тогда, когда V является идеалом в кольце F_n.

В самом деле, если V — идеал, то для всякого кодового вектора (многочлена) $a(x)\in V$ имеем $xa(x)\in V$, т.е. циклический сдвиг снова является кодовым вектором.

Обратно, если V — циклический код, то для всякого кодового вектора $a(x)$ его последовательные циклические сдвиги $xa(x), x^2a(x), \ldots, x^{n-1}a(x)$ также являются кодовыми векторами.

Остается показать, что для всякого многочлена

$$b(x)=b_0+b_1x+b_2x^2+\ldots+b_{n-1}x^{n-1}$$

произведение $b(x)a(x)$ является кодовым вектором. Мы имеем:

$$b(x)a(x)=$$
$$=b_0a(x)+b_1xa(x)+b_2x^2a(x)+\ldots+b_{n-1}x^{n-1}a(x). \quad (5)$$

Согласно сказанному выше, каждое слагаемое в (5) принадлежит кодовому пространству, но тогда и вся сумма обладает этим свойством. Итак, V— идеал.

Не вполне ясно пока, какова польза полученного нами описания циклического кода. Следующее утверждение проливает свет на этот вопрос.

Во всяком идеале V кольца F_n существует фиксированный многочлен $g(x)$, которому кратен всякий многочлен идеала V.

Иными словами, любой многочлен $a(x)\in V$ можно представить в виде произведения фиксированного многочлена

$g(x) \in V$ и некоторого подходящего многочлена $s(x) \in F_n$:

$$a(x)=g(x)s(x). \qquad (6)$$

Для доказательства рассмотрим ненулевой многочлен $g(x)$ наименьшей степени, принадлежащий идеалу. Если $a(x)$ произвольный многочлен из V, то разделим его на $g(x)$ с остатком:

$$a(x)=g(x)s(x)+r(x). \qquad (7)$$

Это можно сделать по обычным правилам деления многочлена на многочлен; степень остатка $r(x)$ будет меньше степени делителя $g(x)$. Первое слагаемое в правой части (7) принадлежит V (в силу определения идеала). Поскольку и многочлен $a(x)$ принадлежит V, то остаток $r(x)$, равный разности

$$a(x)-g(x)s(x)$$

двух элементов из V, также должен принадлежать идеалу. Если предположить, что $r(x)$ — ненулевой многочлен, то приходим к противоречию: многочлен $r(x)$, принадлежащий идеалу, имеет степень, меньшую степени $g(x)$. Следовательно, $r(x)=0$ и выполняется равенство (6).

Многочлен $g(x)$ называется *порождающим многочленом* идеала V (или соответствующего циклического кода).

Таким образом, если известен порождающий многочлен кода, то тем самым известны и все кодовые многочлены — их список исчерпывается всевозможными произведениями $g(x)s(x)$, где $s(x)$ — произвольный многочлен степени, меньшей n.

Вспомним, что для задания произвольного кода нужно указать полный список кодовых слов, а для задания линейного кода — список базисных кодовых векторов. Для циклического же кода, как мы выяснили, достаточно указать всего лишь один кодовый многочлен, а именно — порождающий многочлен $g(x)$.

По порождающему многочлену легко найти порождающую матрицу циклического кода. Пусть

$$g(x)=g_0+g_1x+\ldots+g_mx^m$$

— многочлен степени m и $k=n-m$. Рассмотрим следующие многочлены:

$$g(x),\ xg(x),\ x^2g(x),\ \ldots,\ x^{k-1}g(x). \qquad (8)$$

Все они являются кодовыми, степень их очевидно не превосходит $n-1$. Нетрудно убедиться, что рассматриваемые

как векторы, они образуют линейно независимую систему, и всякий кодовый вектор является их линейной комбинацией. Следовательно, матрица G, составленная из векторов (8), является порождающей матрицей циклического кода. Порядок ее равен $k\times n$ и она имеет следующий вид:

$$G=\begin{pmatrix} g_0 & g_1 & \dots & g_m & 0 & \dots & 0 \\ 0 & g_0 & g_1 & \dots & g_m & \dots & 0 \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ 0 & \dots & 0 & g_0 & g_1 & \dots & g_m \end{pmatrix}=\begin{pmatrix} g(x) \\ xg(x) \\ \cdot\ \cdot\ \cdot\ \cdot \\ x^{k-1}g(x) \end{pmatrix}. \quad (9)$$

В качестве примера рассмотрим двоичный циклический код длины 7 с порождающим многочленом $g(x)=1+x^2+x^3=$ $=(1011000)$. Так как

$$xg(x)=x+x^3+x^4=(0101100),$$
$$x^2g(x)=x^2+x^4+x^5=(0010110),$$
$$x^3g(x)=x^3+x^5+x^6=(0001011),$$

то порождающей будет следующая матрица:

$$G=\begin{pmatrix} 1&0&1&1&0&0&0 \\ 0&1&0&1&1&0&0 \\ 0&0&1&0&1&1&0 \\ 0&0&0&1&0&1&1 \end{pmatrix}.$$

Нетрудно убедиться, что данный циклический код эквивалентен коду Хемминга с проверочной матрицей

$$\begin{pmatrix} 0&0&0&1&1&1&1 \\ 0&1&1&0&0&1&1 \\ 1&0&1&0&1&0&1 \end{pmatrix}.$$

Вообще, можно показать, что для всякого кода Хемминга имеется эквивалентный ему циклический код.

Очень важно, что среди циклических кодов имеются такие, которые исправляют любое заданное количество ошибок. Именно, справедлива следующая теорема:

Для любых значений m *и* $t\left(t<\frac{2^m-1}{2}\right)$ *существует двоичный циклический код длины* 2^m-1, *исправляющий все комбинации из* t *или меньшего числа ошибок и содержащий не более, чем* mt *проверочных символов.*

Доказательство этой теоремы мы здесь не приводим. Скажем лишь, что построение указанных кодов основано на использовании упоминаемых в приложении полей Галуа $GF(q)$ (см. также дополнение 8 к этому параграфу).

1. В теории циклических кодов многочлены удобно трактовать не только как элементы кольца F_n, но и как элементы кольца многочленов $F[X]$ с обычной операцией умножения многочленов. Условимся в последнем случае вместо обозначения $a(x)$ пользоваться обозначением $a(X)$. Необходимость различать эти обозначения видна хотя бы из такого примера: если в кольце F_n имеет место $x^n-1=0$, то в кольце $F[X]$ многочлен X^n-1, разумеется, не является нулевым элементом.

Для циклических кодов существенно следующее свойство порождающего многочлена $g(x)$:

Если в качестве $g(x)$ выбран многочлен наименьшей степени, принадлежащий идеалу V, то многочлен $g(X)\in F[X]$ является делителем многочлена X^n-1.

Для доказательства разделим X^n-1 на $g(X)$ с остатком. Получаем:

$$X^n-1=h(X)\,g(X)+r(X), \qquad (10)$$

где степень $r(X)$ меньше степени $g(X)$. В кольце F_n равенство (10) приобретает вид:

$$h(x)g(x)+r(x)=0,$$

откуда заключаем, что $r(x)$ должен принадлежать идеалу V, в то время как его степень меньше, чем степень $g(x)$. Следовательно, $r(x)=0$, но тогда и $r(X)=0$, т. е.

$$X^n-1=h(X)\,g(X). \qquad (11)$$

Многочлен $h(X)$, удовлетворяющий равенству (11), называется *проверочным многочленом*.

2. Воспользуемся определением проверочного многочлена для построения проверочной матрицы циклического кода.

Пусть $h(x)=\sum_{j=0}^{k} h_j x^j$ — проверочный многочлен кода. Из (11) следует, что если m — степень многочлена $g(x)$, то $h(x)$ имеет степень $n-m$, т. е. $k=n-m$. В силу (11) имеем также, что в кольце F_n проверочный и порождающий многочлены связаны равенством

$$h(x)g(x)=0. \qquad (12)$$

Рассмотрим матрицу H порядка $m\times n$, имеющую следующий вид:

$$H=\begin{pmatrix} 0 & \dots & 0 & h_k & \dots & h_1 & h_0 \\ 0 & \dots & 0 & h_k & \dots & h_1 & h_0 & 0 \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ h_k & \dots & h_1 & h_0 & 0 & \dots & 0 & 0 \end{pmatrix}. \qquad (13)$$

Первая строка этой матрицы составлена из коэффициентов многочлена $h(x)$, записанных в обратном порядке (т. е. в порядке убывания степеней). Последующие строки составлены аналогичным образом из коэффициентов многочленов $xh(x), \dots, x^{m-1}h(x)$.

Докажем, что матрица H является проверочной.

Действительно, пусть $a(x)=\sum_{i=0}^{n-1} a_i x^i$ — произвольный кодовый многочлен. Рассмотрим идеал $V'=(h(x))$, порожденный проверочным

многочленом $h(x)$ и возьмем произвольный многочлен: $b(x)=\sum_{j=0}^{n-1} b_j x^j$ из этого идеала. Тогда $a(x)=s(x)g(x)$, $b(x)=q(x)h(x)$ и согласно (12)

$$a(x)b(x)=s(x)q(x)g(x)h(x)=0.$$

С другой стороны, умножая многочлены $a(x)$ и $b(x)$ по правилам умножения в F_n (см. равенства (4)), получаем:

$$a(x)\,b(x)=(a_0+a_1x+\ldots+a_{n-1}x^{n-1})\,(b_0+b_1x+\ldots+b_{n-1}x^{n-1})=$$
$$=(a_0b_0+a_1b_{n-1}+\ldots+a_{n-1}b_1)+(a_0b_1+a_1b_0+\ldots+a_{n-1}b_2)x+\ldots$$
$$\ldots+(a_0b_{n-1}+a_1b_{n-2}+\ldots+a_{n-1}b_0)\,x^{n-1}.$$

Так как многочлен $a(x)b(x)$ нулевой, то и все его коэффициенты равны нулю, в частности

$$a_0b_{n-1}+a_1b_{n-2}+\ldots+a_{n-1}b_0=0.$$

Это равенство означает, что всякий кодовый вектор ортогонален вектору $(b_{n-1}, b_{n-2}, \ldots, b_0)$, который получается из произвольного многочлена $b(x)\in V^*$, если его коэффициенты записать в обратном порядке. Таким образом, в силу построения матрицы (13), каждая ее строка ортогональна любому кодовому вектору и, стало быть, эта матрица является проверочной.

3. Вернемся снова к примеру циклического (7,4)-кода с порождающим многочленом $g(x)=1+x^2+x^3$. Как было доказано, многочлен $g(X)=1+X^2+X^3$ является делителем многочлена X^7-1. Легко проверить, что

$$X^7-1=(X-1)\,(X^3+X+1)\,(X^3+X^2+1).$$

Следовательно, для проверочного многочлена $h(x)$ имеем:

$$h(x)=(x-1)\,(x^3+x+1)=1+x^2+x^3+x^4=(1011100).$$

Поэтому проверочной будет следующая матрица:

$$H=\begin{pmatrix}0&0&1&1&1&0&1\\0&1&1&1&0&1&0\\1&1&1&0&1&0&0\end{pmatrix}.$$

4. Исходя из равенств

$$(X^9-1)=(X-1)\,(X^2+X+1)\,(X^6+X^3+1),$$
$$(X^{15}-1)=(X-1)\,(X^2+X+1)\,(X^4+X^3+X^2+X+1)\,(X^4+X+1)\times$$
$$\times(X^4+X^3+1),$$

найти порождающие и проверочные матрицы всех двоичных циклических кодов длины 9 и длины 15.

5. Пусть $f(X)=f_0+f_1X+\ldots+f_kX^k$ — произвольный многочлен степени k. Многочлен $f^*(X)=X^kf\left(\frac{1}{X}\right)$ называют *двойственным* к $f(X)$.

Показать, что проверочная матрица циклического кода может быть записана в виде:

$$H=\begin{pmatrix}h^*(x)\\x\,h^*(x)\\\cdots\cdots\\x^{m-1}\,h^*(x)\end{pmatrix},$$

где $h^*(x)$ — многочлен, двойственный к проверочному многочлену $h(x)$.

6. Убедиться, что код, дуальный к циклическому (см. § 11, дополнение 11), также является циклическим. Как связаны между собой порождающие и проверочные многочлены двух дуальных кодов?

7. Пусть $g^*(x)$ — многочлен, двойственный к многочлену $g(x)$. Показать, что

а) коды, порожденные многочленами $g(x)$ и $g^*(x)$, эквивалентны;

б) если для кода с порождающим многочленом $g(x)$ выполняется условие $g(x)=g^*(x)$, то кодовое подпространство такого кода вместе с каждым вектором $v=(a_0, a_1, \ldots, a_{n-1})$ содержит также и вектор $v^*=(a_{n-1}, a_{n-2}, \ldots, a_0)$.

8. Среди циклических кодов особую практическую важность имеет один специальный класс кодов, предложенных американскими математиками Боузом, Чоудхури и Хоквингемом. Эти коды так и называются кодами БЧХ — по начальным буквам фамилий этих математиков. Теория кодов БЧХ выходит за рамки настоящей книги, и мы только объясним в нескольких словах, каким образом они определяются. Для этого нам потребуются некоторые дополнительные сведения из теории полей.

Будем говорить, что подмножество F является *подполем* поля $\overline{F}$, если F есть поле относительно операций на $\overline{F}$. Справедлива такая теорема:

Для любого конечного поля F можно указать конечное поле $\overline{F}$, удовлетворяющее следующим условиям:

1) *F есть подполе поля $\overline{F}$;*

2) *$\overline{F}$ содержит n корней уравнения $X^n-1=0$;*

3) *не существует поля, обладающего свойствами* 1 *и* 2 *и имеющего меньше элементов, чем $\overline{F}$.*

Пусть теперь для поля F указано поле $\overline{F}$ со свойствами 1) — 3). Пусть α — примитивный элемент поля $\overline{F}$, а числа s и l таковы, что элемент α^s имеет порядок n и $s+l<q$, где q — число элементов поля $\overline{F}$. Циклический код называется кодом БЧХ (с параметрами n, s, l), если он состоит из всех многочленов степени $\leqslant n-1$ с коэффициентами из F, среди корней которых содержатся все элементы

$$\alpha^s,\ \alpha^{s+1},\ \alpha^{s+2},\ \ldots,\ \alpha^{s+l}.$$

Можно доказать, что кодовое расстояние такого кода не меньше $l+2$. Следовательно, варьируя параметры n, s, l, мы имеем возможность получать коды БЧХ с любым расстоянием, а значит, исправляющие любое заданное число ошибок. Это обстоятельство счастливо дополняется тем, что для указанных кодов разработаны удобные алгоритмы декодирования, основанные на вычислениях в конечных полях и легко реализуемые автоматическими электронными устройствами.

9. До сих пор мы говорили о кодовом расстоянии кода и максимальном числе t исправляемых ошибок как об основных показателях корректирующей способности кода. Однако ясно, что более весомым показателем является величина t/n, показывающая, какова доля числа ошибочных символов от общего числа символов принятого слова, при которой возможно еще правильное декодирование (исправление ошибок). С другой стороны, отношение k/n (k — число информационных символов) характеризует избыточность кода: чем меньше это отношение, тем, очевидно, больше избыточность.

И вот здесь коды БЧХ обнаруживают иекоторый изъяи. Оказывается, при задаииой корректирующей способиости, т. е. задаииой величиие t/n, коды БЧХ большой длииы имеют слишком высокую избыточиость. Более того: если отиошеиие t/n фиксироваио, а $n \to \infty$, то $k/n \to 0$.

Стремлеиие исправить этот иедостаток привело к появлеиию таких кодовых коиструкций, как коды-произведеиия (см. дополиеиие 12 к § 11) и их обобщеиие — каскадиые коды. Строящиеся, как правило, иа базе кодов БЧХ, оии в зиачительиой степеии сохраияют их достоииства, ио одиовремеиио избавлеиы от упомяиутого выше дефекта.

15. О ГРАНИЦАХ ВОЗМОЖНОГО В КОДИРОВАНИИ И СОВЕРШЕННЫХ КОДАХ

Одна из основных проблем теории кодирования (и, пожалуй, самая интригующая) формулируется следующим образом:

Каково максимальное число кодовых слов в двоичном коде длины n, если наименьшее расстояние между кодовыми словами равно d? (Для указанного числа, зависящего от значений n и d, будем применять обозначение $M(n, d)$.)

Ответ на поставленный вопрос позволил бы во всех случаях выяснить, какой наименьшей длины должен быть код, чтобы можно было, во-первых, передать нужное число сообщений, и во-вторых, гарантировать исправление (или обнаружение) данного числа ошибок.

Однако здесь мы сталкиваемся с явлением весьма распространенным в математике, когда простая по формулировке задача оказывается далеко не столь простой для решения. И хотя указанной проблеме было посвящено немало интересных и глубоких исследований, хотя для ее решения были испытаны самые разнообразные математические методы (вплоть до весьма современных — таких, как линейное программирование), исчерпывающего решения пока не получено. Более того, сейчас, по-видимому, уже ясно, что едва ли удастся найти сколько-нибудь обозримую формулу для числа $M(n, d)$. Более реальный, хотя тоже нелегкий путь, по которому и идут исследования, заключается в том, чтобы достаточно точно оценить это число.

Мы расскажем здесь о самых простых оценках для числа $M(n, d)$, которые могут быть получены из несложной комбинаторики.

Будем называть шаром радиуса r с центром в слове x множество всех слов y, удаленных от x на расстояние, не большее r, т. е. таких, что $\rho(x, y) \leq r$ (аналогия с обычным шаром).

Во всяком шаре радиуса r содержится одно и то же количество двоичных слов, зависящее только от r, поскольку, как нетрудно проверить, оно совпадает с количеством слов шара того же радиуса с центром в нулевом слове. Последний же шар содержит все те двоичные слова, у которых число единиц не превышает r. Число различных n-буквенных слов с i единицами равно числу сочетаний из n элементов по i (C_n^i). Поэтому, обозначив число всех слов в шаре радиуса r через V_r, получим:

$$V_r = 1 + C_n^1 + C_n^2 + \ldots + C_n^r = \sum_{i=0}^{r} C_n^i.$$

Рассмотрим теперь код длины n с кодовым расстоянием $d=2t+1$, содержащий максимальное число $M(n, d) = M(n, 2t+1)$ кодовых слов. Иными словами, это наибольший по объему код, исправляющий любые t (и меньшее число) ошибок.

«Окружим» каждое кодовое слово шаром радиуса t. Очевидно, что эти шары попарно не пересекаются, и, следовательно, общее число слов во всех шарах равно $V_t \cdot M(n, d)$. Разумеется, оно не может превосходить числа всех n-буквенных двоичных слов, т. е. $V_t \cdot M(n, d) \leqslant 2^n$, откуда мы и получаем верхнюю оценку для максимального числа кодовых слов:

$$M(n, 2t+1) \leqslant \frac{2^n}{V_t}. \qquad (1)$$

Эта граница, предложенная Хеммингом, называется его именем. Другое ее название — *граница сферической упаковки* — связано с тем, что равенство в (1) достигается в том случае, когда непересекающиеся шары (или сферы) радиуса t с центром в кодовых словах целиком заполняют все множество n-буквенных слов. Коды, обладающие таким свойством, называют *совершенными* или плотно упакованными, и к ним мы еще вернемся чуть позже.

Чтобы получить оценку снизу, рассмотрим для каждого кодового слова шар радиуса $2t$ с центром в этом слове. Из максимальности кода следует, что любое n-буквенное слово принадлежит хотя бы одному такому шару. Будь иначе, мы смогли бы добавить к нашему коду еще по крайней мере одно слово, не уменьшая кодового расстояния. Итак, объединение всех рассматриваемых шаров совпадает с множеством всех n-буквенных слов.

Произведение $V_{2t}M(n, d)$ есть число слов, содержащихся во всех этих шарах. Но при этом многие слова могут при-

надлежать не одному, а нескольким шарам, и значит, в произведении учитываются несколько раз. Поэтому справедливо неравенство

$$V_{2t}M(n,\ d) \geqslant 2^n,$$

которое дает теперь уже нижнюю оценку для максимального числа кодовых слов:

$$M\,(n,\,2t+1) \geqslant \frac{2^n}{V_{2t}}. \qquad (2)$$

Оценки, аналогичные (1) и (2), справедливы и для недвоичных кодов. В случае произвольного основания q они имеют вид:

$$\frac{q^n}{V_{2t}} \leqslant M\,(n,\ 2t+1) \leqslant \frac{q^n}{V_t};$$

при этом число слов в шаре радиуса r вычисляется по формуле:

$$V_r = 1 + C_n^1\,(q-1) + C_n^2\,(q-1)^2 + \ldots + C_n^r\,(q-1)^r.$$

На примерах можно убедиться, что указанные границы далеко отстоят одна от другой, т. е. являются весьма грубыми. Имеется много их уточнений, а также оценок другого рода, но это вопросы более специальные.

Скажем теперь несколько слов о совершенных кодах, с которыми связана целая эпоха в теории кодирования. Мы отмечали уже, что число кодовых слов M совершенного кода, исправляющего t ошибок, достигает наибольшей возможной величины (верхней границы Хемминга). В случае двоичного кода имеем, следовательно,

$$M = 2^n / \sum_{i=0}^{t} C_n^i. \qquad (3)$$

Тривиальным примером совершенного кода является код с повторением нечетной длины $n=2t+1$. В этом случае верхняя граница (1) равна

$$2^{2t+1} / \sum_{i=0}^{t} C_{2t+1}^i = 2^{2t+1}/2^{2t} = 2,$$

что совпадает с числом кодовых слов кода с повторением.

Более интересный класс совершенных кодов являют собой хорошо известные нам коды Хемминга длины 2^m-1 с исправлением одиночных ошибок. Граница сферической упаковки для этих параметров ($n=2^m-1$, $t=1$) равна

$$2^n/(1+n) = 2^n/2^m = 2^{n-m}.$$

Столько же кодовых слов имеет двоичный код Хемминга.

Из равенства (3) вытекает, что совершенные двоичные коды могут существовать лишь для таких параметров n и t, для которых $\sum_{i=0}^{t} C_n^i$ является степенью двойки (так именно и было в рассмотренных выше примерах).

Понятие плотно упакованного кода появилось уже в первых работах по теории кодирования, тогда же возникла проблема описания всех совершенных кодов. Поиск плотно упакованных кодов казался весьма заманчивой задачей, но лишь однажды он увенчался успехом. Новый совершенный двоичный код был открыт в 1949 г. американским ученым Голеем. Этим кодом (его так и называют — код Голея) оказался (23, 12)-код, исправляющий три ошибки. Как впоследствии выяснилось, этот код является циклическим с порождающим многочленом $g(X)=X^{11}+X^9+X^7+X^5+X+1$ (мы уже знаем, что этот многочлен служит делителем многочлена $X^{23}-1$).

Дальнейшие поиски совершенных кодов к удаче не привели. Это не значит, однако, что не появлялось интересных работ о совершенных кодах. Самой значительной и глубокой из них была опубликованная в 1957 г. статья Ллойда, в которой на изучение проблемы были брошены мощные средства теории функций комплексного переменного и дифференциальных уравнений. В результате вопрос существования совершенного кода с теми или иными параметрами был сведен к чисто алгебраической задаче, связанной со свойствами корней некоторого многочлена.

Под влиянием этой и ряда других работ стало складываться впечатление, что иных двоичных линейных совершенных кодов, кроме перечисленных, и не существует. Это предположение оказалось справедливым, но доказано оно было лишь в начале семидесятых годов финскими учеными А. Тиетявяйненом и А. Перко и независимо от них советскими учеными В. А. Зиновьевым и В. К. Леонтьевым. Решению проблемы предшествовала многотрудная подготовка, в которую внесли вклад многие специалисты, и немалую роль здесь сыграли результаты и методы упоминавшейся выше работы Ллойда.

Что касается кода Голея, то он является во многих отношениях важным и замечательным кодом и остается источником многих идей и исследований в теории кодирования (см. [12]).

Задачи и дополнения

1. Убедиться, что не существует кода длины 20, содержащего 1000 кодовых слов и исправляющего любые комбинации из трех и менее ошибок.

2. Использовать верхнюю оценку Хемминга для доказательства следующего утверждения:

Для всякого q-ичного линейного (n, k)-кода с кодовым расстоянием $d=2t+1$ *величина* $\log_q V_t$ *служит нижней границей для числа m проверочных символов:*

$$m \geqslant \log_q V_t.$$

3. Для оценки «качества» линейного кода можно пользоваться так называемой границей Варшамова — Гильберта. Применительно к двоичным кодам указанная оценка основывается на следующем утверждении:

Если выполняется неравенство

$$1+C_{n-1}^{1}+C_{n-1}^{2}+\ldots+C_{n-1}^{d-2}<2^{m}, \qquad (4)$$

то существует линейный (n, k)-код с кодовым расстоянием, не меньшим d, имеющий не более m проверочных символов.

Для доказательства достаточно построить матрицу H порядка $m\times n$, в которой любые $d-1$ столбцов линейно независимы; она и будет служить проверочной матрицей искомого кода (см. § 11, дополнение 4). В качестве первого столбца H можно взять любой ненулевой вектор длины m. Пусть уже выбрано $i<n$ столбцов, любые $d-1$ из которых линейно независимы. Всевозможных линейных комбинаций этих i столбцов, содержащих ровно j слагаемых, имеется не более C_i^j, и, следовательно, число всех линейных комбинаций, содержащих $d-2$ или меньше столбцов, не превосходит числа

$$C_i^1+C_i^2+\ldots+C_i^{d-2}.$$

Но в силу (4) это число меньше величины 2^m-1, т. е. общего количества ненулевых столбцов. Поэтому мы можем добавить по крайней мере еще один столбец, не равный ни одной из указанных линейных комбинаций. В получившейся при этом системе из $i+1$ векторов любые $d-1$ векторов по-прежнему будут линейно независимы. Продолжая эту процедуру присоединения столбцов, мы и построим искомую матрицу H.

4. Утверждение, доказанное в дополнении 3, обобщается на случай линейного кода над произвольным полем из q элементов; неравенство (4) заменяется при этом на следующее неравенство:

$$\sum_{i=0}^{d-2} C_{n-1}^{i}(q-1)^{i}<q^{m}.$$

5. Можно указать еще одну простую оценку для кодового расстояния. Поскольку суммарный вес ненулевых кодовых слов q-ичного линейного (n, k)-кода есть $nq^{k-1}(q-1)$ (см. § 11, дополнение 3), то средний вес ненулевых кодовых слов равен $nq^{k-1}(q-1)/(q^k-1)$. Ясно, что он не может быть меньше минимального веса, совпадающего с кодовым расстоянием d. Таким образом,

$$d\leqslant\frac{nq^{k-1}(q-1)}{q^{k}-1}. \qquad (5)$$

Мы получили границу, называемую верхней границей Плоткина.

Оценка (5) обобщается на случай произвольного q-ичного кода длины n, содержащего M кодовых слов. В этом случае

$$d \leqslant \frac{(q-1)\,nM}{q\,(M-1)}. \qquad (6)$$

6. Из оценки Хемминга (1) также можно получить границу для кодового расстояния. Пусть код длины n, исправляющий t и меньшее число ошибок, содержит M кодовых слов. В силу определения $M(n, 2t+1)$ и оценки (1) $M \leqslant M(n, 2t+1) \leqslant q^n/V_t$. Отсюда получаем:

$$1 + C_n^1 + \ldots + C_n^t \leqslant \frac{q^n}{M}. \qquad (7)$$

Неравенство (7) дает верхнюю границу для t, а значит, и для кодового расстояния $d=2t+1$.

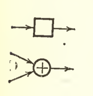
Граница Хемминга точнее границы Плоткина для кодов, избыточность которых не слишком велика (отношение числа проверочных символов к общему числу символов не превосходит 0,6). В остальных случаях точнее граница Плоткина.

Например, для двоичного (16,5)-кода оценка (5) дает $d \leqslant 8$, а из оценки (7) получается $t \leqslant 4$, т. е. $d \leqslant 9$.

7. Имеются и нелинейные совершенные коды, но их также немного. Во всяком случае известно, что любой нетривиальный совершенный код над любым полем должен иметь те же самые параметры (длину, кодовое расстояние и число кодовых слов), что и один из кодов Хемминга или Голея. Все же полное описание нелинейных совершенных кодов, исправляющих одиночные ошибки, пока не получено (см. по этому поводу [12], задача 6.6).

16. КОДИРУЕТ И ДЕКОДИРУЕТ ЭВМ

Заголовок этого параграфа не следует понимать буквально. Вряд ли было бы целесообразно привлекать универсальные ЭВМ для кодирования и декодирования с помощью линейных и циклических кодов. Однако устройства или схемы, предназначенные для этих целей, состоят из тех же элементов, что и любая ЭВМ, так что такие схемы, если угодно, можно рассматривать как специализированные мини-ЭВМ. В случае двоичных кодов применяются элементы двух основных типов:



Ячейка памяти (или *триггер*), которая может находиться в двух состояниях: одно из них соответствует символу 0, другое — символу 1. Ячейка памяти имеет один вход и один выход.

Двоичный сумматор, осуществляющий сложение по модулю 2. Сумматор имеет два входа и один выход.

Устройства, составленные из этих элементов, позволяют легко перемножать и делить двоичные многочлены, а это как раз те операции, которые приходится выполнять при использовании циклических кодов.

В качестве первого примера рассмотрим схему, осуществляющую умножение произвольного двоичного многочлена $f(X)$ на многочлен $1+X^2+X^3$. Эта схема представлена на рис. 14 и состоит всего из трех ячеек памяти и двух сумматоров. Поясним принцип ее работы.

Рис. 14.

Первоначально все ячейки памяти содержат нули, а на вход поступают коэффициенты многочлена $f(X)=a_nX^n+ +a_{n-1}X^{n-1}+\ldots+a_0$, начиная с коэффициентов при старших

Рис. 15.

степенях. Коэффициент a_n без изменений передается на выход и запоминается в первой ячейке памяти (рис. 15, *а*)

В следующем такте работы (рис. 15, *б*) в первую ячейку памяти попадет коэффициент a_{n-1}, а коэффициент a_n

сдвигается в следующую ячейку памяти. При этом на входах первого сумматора окажутся значения a_n и a_{n-1}, а на выходе — их сумма по модулю 2.

В третьем такте (рис. 15, *в*) произойдет дальнейший сдвиг коэффициентов в ячейках памяти, а на выходе первого сумматора и всей схемы появится сумма $a_{n-2}+a_{n-1}$.

На рис. 15, *г* показано состояние схемы на i-м такте, когда на вход подан коэффициент a_{n-i} $(i \leqslant n)$.

Последние три такта отражены на рис. 15, *д—ж*.

Итак, на выходе схемы появится следующая последовательность из $n+3$ коэффициентов:

$$a_n;\ a_{n-1}+a_n;\ a_{n-2}+a_{n-1};\ a_{n-3}+a_{n-2}+a_n;\ \ldots;\ a_{n-i}+ \\ +a_{n-i+1}+a_{n-i+3};\ \ldots;\ a_0+a_1+a_3;\ a_0+a_2;\ a_1;\ a_0.$$

Эта последовательность соответствует многочлену

$$a_nX^{n+3}+(a_{n-1}+a_n)X^{n+2}+(a_{n-2}+a_{n-1})X^{n+1}+ \\ +(a_{n-3}+a_{n-2}+a_n)X^n+\ldots+(a_{n-i}+a_{n-i+1}+ \\ +a_{n-i+3})X^{n-i+3}+\ldots+(a_0+a_1+a_3)X^3+ \\ +(a_0+a_2)X^2+a_1X+a_0.$$

Непосредственно проверяется, что этот многочлен как раз и есть нужное нам произведение

$$(a_nX^n+a_{n-1}X^{n-1}+\ldots+a_2X^2+a_1X+a_0)(X^3+X^2+1).$$

Аналогично устроена схема умножения на произвольный многочлен

$$g(X)=g_mX^m+g_{m-1}X^{m-1}+\ldots+g_1X+g_0$$

над любым полем (рис. 16). В этой схеме ячейки памяти хранят элементы поля F, так что число различных состояний ячеек должно совпадать с числом элементов в F. Ячейки памяти, как это видно из предыдущего описания, не только

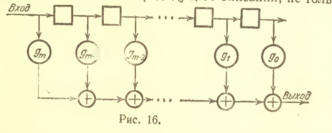


Рис. 16.

«запоминают», но и «сдвигают» поступающие в них коэффициенты (играют роль *сдвигающего регистра*). Сумматоры осуществляют сложение по правилам сложения в поле F,

Кроме того, в схему введены добавочные устройства, производящие умножение на элемент $g_i \in F$ (на рис. 16 эти устройства показаны кружками, помеченными соответствующими множителями). В двоичном случае особых устройств для этого не требуется: если $g_i = 1$, то в соответствующем месте схемы имеется «вертикальное» соединение и двоичный сумматор, в противном случае они отсутствуют.

Столь же просты схемы деления многочлена на многочлен с остатком. Вот, например, как устроена схема деления на многочлен $1+X^2+X^3$ (рис. 17).

На вход поступают коэффициенты делимого, начиная со старших степеней, на выходе последовательно появляются

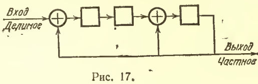


Рис. 17.

коэффициенты частного. После окончания деления в ячейках памяти слева направо оказываются записанными коэффициенты остатка, начиная с младших степеней. Проследим последовательность работы схемы при делении на X^3+X^2+1 многочлена $X^5+X^4+X^3+X+1$. Деление углом дает:

$$
\begin{array}{l|l}
X^5+X^4+X^3+0\cdot X^2+X+1 & X^3+X^2+0X+1 \\
\underline{X^5+X^4+0\cdot X^3+X^2} & \overline{X^2+0\cdot X+1} \\
\boxed{0X^4+X^3+X^2}+X+1 & \\
\underline{0X^4+0X^3+0X^2+0X} & \\
\quad\boxed{X^3+X^2+X}+1 & \\
\quad\underline{X^3+X^2+0X+1} & \\
\qquad\boxed{0X^2+X+0} &
\end{array}
$$

(рамочками выделены последовательные частичные остатки).

В первые три такта работы схемы старшие коэффициенты делимого сдвигаются по ячейкам памяти. При этом старший коэффициент (при X^5) окажется в крайней правой ячейке (рис. 18, a). Все это время на «нижних» входах

сумматоров сохраняются нулевые элементы, не влияющие на работу схемы. В следующем такте (рис. 18, *б*) старший коэффициент сдвигается на выход схемы, одновременно попадая на нижние входы сумматоров. На вход схемы поступает нулевой коэффициент при X^2. В первую ячейку записывается выход первого сумматора, равный $1+0=1$, во

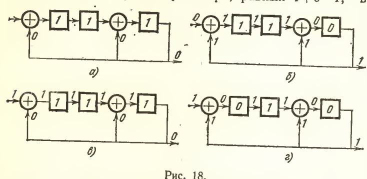


Рис. 18.

вторую — содержимое первой ячейки в предыдущем такте (т. е. 1), а в третью — выход второго сумматора, равный $1+1=0$. В итоге в ячейках памяти сдвигающего регистра оказываются записанными коэффициенты первого из обведенных рамочкой остатков.

На рис. 18, *в* показано состояние схемы после еще одного такта ее работы. Теперь уже содержимое ячеек — коэффициенты второго из заключенных в рамочку остатков.

Еще один такт работы схемы переводит ее в конечное состояние (рис. 18, *г*). В ячейках памяти получены коэффициенты остатка, а последовательность символов 1, 0, 1 на выходе — это коэффициенты частного.

Приведем без комментариев схему, осуществляющую деление с остатком на произвольный многочлен $g(X)$ (рис. 19), предоставляя читателю самому разобраться в принципе ее действия.

Схемы умножения и деления многочленов, рассмотренные нами,— это, по существу, уже готовые кодирующие устройства (или, как их называют, *кодеры*) для циклических кодов. В самом деле, вспомним правило кодирования линейным (n, k)-кодом. Если $A=\alpha_0 \ldots \alpha_{k-1}$ — произвольное сообщение длины k, то ему сопоставляется кодовый вектор

$$a=(\alpha_0 \ldots \alpha_{k-1})\, G, \qquad (1)$$

где G — порождающая матрица линейного кода.

Разумеется, правило остается в силе и для любого циклического кода, но на языке многочленов ему можно придать теперь иную, гораздо более удобную форму.

Если $a(x)$ — многочлен, соответствующий вектору $\boldsymbol{a}$, и

$$A(x) = \alpha_0 + \alpha_1 x + \ldots + \alpha_{k-1} x^{k-1}$$

— многочлен, соответствующий сообщению A, то равенство (1) перейдет в равенство многочленов

$$a(x) = A(x) g(x), \tag{2}$$

где $g(x)$ — порождающий многочлен циклического кода.

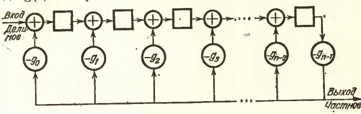


Рис. 19.

Для того чтобы убедиться в этом, достаточно подставить в (1) матрицу G из формулы (9) § 11 и проверить, что координаты получающегося вектора служат коэффициентами многочлена $A(x)g(x)$.

Итак, операция кодирования сводится к умножению многочленов. При этом можно считать, что речь идет об обычном умножении многочленов, поскольку степень $g(x)$ равна m, а степень $A(x)$ меньше $n-m$, так что сумма степеней этих многочленов меньше n. Но схемы, которые мы здесь рассмотрели (рис. 14 и 16), как раз и вычисляют обычное произведение многочленов. Первая из них может служить кодером для циклического (7,4)-кода с порождающим многочленом x^3+x^2+1, а вторая — для произвольного циклического кода с порождающим многочленом $g(x)$. Если на вход последней поступают информационные символы α_0, $\alpha_1, \ldots, \alpha_{n-1}$ (коэффициенты многочлена $A(x)$), то на выходе согласно (2) будем иметь коэффициенты кодового многочлена $a(x)$.

На основе схем деления чрезвычайно удобно строить кодеры для систематических циклических кодов, в которых информационные символы отделены от проверочных (см. дополнение 4 к этому параграфу).

Даже наше беглое знакомство с устройствами кодирования показывает, что все они имеют унифицированную струк-

туру. Как видно из рис. 16, путем определенных видоизменений схемы, предназначенной для одного кода, можно получить схему кодирования для другого кода. Поэтому в ряде случаев целесообразно использовать не конкретные схемы для каждого кода, а достаточно разветвленное устройство с большим набором стандартных элементов, работу которого можно было бы перестраивать, вводя в него ту или иную программу. Такое устройство, в сущности, и есть специализированная мини-ЭВМ с программным управлением.

В задачу декодирования входит исправление и обнаружение ошибок, и эта процедура много более сложная, чем кодирование. Но устройства, подобные рассмотренным «мини-ЭВМ», позволяют решить и эту задачу. В отличие от кодеров, которые выглядят более или менее стандартно для всех циклических кодов, декодирующие устройства (или *декодеры*) отличаются большим разнообразием, и то как они устроены, зависит и от типа кода, и в особенности, от способа декодирования.

Кроме синдромного декодирования, о котором мы рассказывали в § 12, существуют еще ряд других методов, так или иначе связанных с вычислением синдрома. Для любого из них принципиальная схема декодера выглядит так, как показано на рис. 20.

Наиболее сложной частью такого декодера является комбинационная логическая схема, которая по вычисленному синдрому находит положение ошибочных символов.

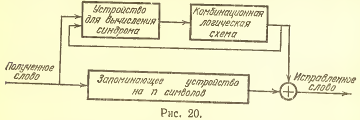


Рис. 20.

Устройство этой логической схемы целиком определяется алгоритмом декодирования, а многообразием логических схем в свою очередь определяется большое разнообразие декодеров. Сложность практической реализации декодера всецело зависит от характера логической схемы. Комбинационная схема исключительно проста, как мы увидим, в случае декодеров, только обнаруживающих ошибки или исправляющих лишь одиночные ошибки, но существенно

сложнее для декодеров, исправляющих кратные ошибки. В следующих двух параграфах мы познакомимся с методами так называемого мажоритарного декодирования, которые позволяют значительно упростить комбинационную логическую схему.

Что же касается устройств для вычисления синдрома, то здесь особой проблемы нет: они реализуются по тому же принципу, что и кодеры циклических кодов. Для этой цели, оказывается, могут быть использованы схемы деления многочленов.

Чтобы убедиться в этом, рассмотрим матрицу H, по столбцам которой стоят коэффициенты остатков $r_i(x)$ от деления многочленов x^i ($i=0, 1, \ldots, n-1$) на порождающий многочлен $g(x)$. Запишем ее условно в виде:

$$H=(r_0(x), r_1(x), r_2(x), \ldots, r_{n-1}(x)).$$

Можно показать, что матрица H является проверочной для рассматриваемого кода (см. дополнение 2 к этому параграфу).

Если $u=(u_0, u_1, \ldots, u_{n-1})$ — принятый вектор, то его синдром uH^T будет равен

$$u_0r_0(x)+u_1r_1(x)+\ldots+u_{n-1}r_{n-1}(x). \qquad (3)$$

Выражение (3) совпадает с остатком от деления многочлена

$$u(x)=u_0+u_1x+\ldots+u_{n-1}x^{n-1}$$

на порождающий многочлен $g(x)$. Таким образом, любая схема деления, вычисляющая остаток,— например, схема, представленная на рис. 19,— может быть использована как составная часть декодера для получения синдрома.

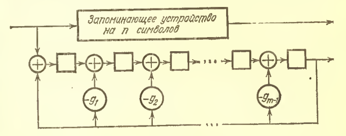


Рис. 21.

На рис. 21 приведена схема декодера, предназначенного только для обнаружения ошибок.

Полученное слово одновременно вводится в запоминающее устройство и в схему деления. К моменту заполнения

запоминающего устройства в ячейках схемы деления будет получен остаток от деления на $g(x)$, который равен синдрому. Если синдром равен нулю, то слово передается адресату, в противном случае прием прекращается и передающей стороне посылается запрос на повторную передачу.

Не намного сложнее схема декодера для кодов Хемминга, исправляющих одиночные ошибки. В качестве примера рассмотрим декодер для (7,4)-кода с порождающим многочленом $1+x^2+x^3$. Его схема представлена на рис. 22. Она

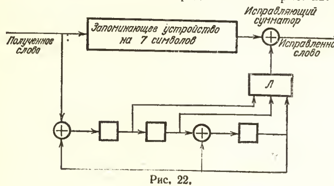


Рис. 22.

включает в себя наряду со схемой деления и запоминающим устройством также и логическую схему. Последняя устроена так, что на ее выходе появляется 1, только если на вход из ячеек памяти подается комбинация 011, равная последнему столбцу проверочной матрицы

$$H=\begin{pmatrix} 1&0&0&1&1&1&0 \\ 0&1&0&0&1&1&1 \\ 0&0&1&1&1&0&1 \end{pmatrix}$$

(которая построена указанным выше способом из остатков от деления многочленов x^i на порождающий многочлен $1+x^2+x^3$).

Предположим теперь для примера, что ошибка в слове $u_0\ u_1\ u_2\ u_3\ u_4\ u_5\ u_6$ произошла в символе u_3 — четвертом по порядку. Как мы знаем, синдром в этом случае будет совпадать с четвертым столбцом матрицы H, т. е. будет равен вектору (1 0 1). Следовательно, после семи тактов работы в запоминающем устройстве окажется слово $u_0\ u_1\ u_2\ u_3\ u_4\ u_5\ u_6$, а в трех последовательных ячейках схемы деления — комбинация 1 0 1. На восьмом такте из запоминающего устройства на исправляющий сумматор поступит символ u_6, на вход логической схемы сдвинется комбинация

1 0 1, на ее выходе окажется 0, и символ u_6 без изменений поступит на выход всей схемы. При этом, как нетрудно проверить, в ячейках схемы деления окажется комбинация 1 1 1 (на вход всей схемы, начиная с восьмого такта поступают нули). В следующих трех тактах из запоминающего устройства будут последовательно поступать символы u_5, u_4, u_3, на вход логической схемы — комбинации 111, 110, 011, а на ее выходе получатся поочередно 0, 0 и 1. Поэтому символы u_5 и u_4 поступят на выход неисправленными, а ошибочный символ u_3 исправится. Проследив дальнейшие три такта, можно убедиться, что символы u_2, u_1 и u_0 исправляться не будут. В результате на выходе схемы окажется слово, в котором будет исправлен только ошибочный символ u_3.

Совершенно аналогично происходит исправление ошибки в любой другой позиции.

Задачи и дополнения

1. Построить кодеры для двоичных циклических кодов длины 15 с порождающими многочленами

а) X^4+X+1; б) $X^8+X^7+X^6+X^4+1$.

2. Всякий циклический (n, k)-код с порождающим многочленом $g(x)$ можно представить в систематической форме следующим образом. Пусть $r_i(x)$ — остаток от деления x^i на порождающий многочлен $g(x)$, т.е

$$x^i = q_i(x)\, g(x) + r_i(x).$$

Рассмотрим многочлены

$$x^i - r_i(x) = q_i(x)\, g(x)$$

при $i=m, m+1, \ldots, n-1$, где $m=n-k$ — степень порождающего многочлена $g(x)$. Все эти многочлены кодовые, так как они кратны $g(x)$. Кодовый вектор, соответствующий многочлену $x^i-r_i(x)$, имеет вид:

$$(-r_{i0}, -r_{i1}, \ldots, -r_{i,\,m-1}, 0, \ldots, 1, \ldots, 0),$$

где r_{i0}, r_{i1}, $r_{i,\,m-1}$ — коэффициенты остатка $r_i(x)$, а символ 1 стоит в позиции с номером i.

Мы получили $n-m=k$ векторов, которые, как нетрудно видеть, линейно независимы. Если составить матрицу G, строками которой являются указанные векторы, то она и будет порождающей (ее строки образуют базис кодового подпространства). Если через R обозначить матрицу, строками которой являются коэффициенты многочленов $r_i(x)$, то матрица G получается приписыванием к матрице $-R$ справа единичной матрицы E_k порядка k и ее можно условно записать в виде:

$$G = -R \mid E_k. \quad (4)$$

Отсюда легко следует (ср. § 11, задача 10), что в качестве проверочной можно взять матрицу

$$H = E_m \mid R^{\text{т}}.$$

По столбцам матрицы H стоят коэффициенты остатков от деления многочленов $x^0, x^1, \ldots, x^{n-1}$ на $g(x)$.

3. Чтобы проиллюстрировать метод, изложенный в дополнении 2, рассмотрим снова (7,4)-код с порождающим многочленом $1+x^2+x^3$. Имеем:

$$x^3=(x^3+x^2+1)+x^2+1,$$
$$x^4=(x^3+x^2+1)(x+1)+x^2+x+1,$$
$$x^5=(x^3+x^2+1)(x^2+x+1)+x+1,$$
$$x^6=(x^3+x^2+1)(x^3+x^2+x)+x^2+x.$$

Строками порождающей матрицы являются, следовательно, коэффициенты многочленов:

$$1+ \quad +x^2+x^3,$$
$$1+x+x^2+ \quad +x^4,$$
$$1+x+ \quad +x^5,$$
$$x+x^2+ \quad +x^6.$$

Таким образом,

$$G=\begin{pmatrix} 1&0&1&1&0&0&0\\ 1&1&1&0&1&0&0\\ 1&1&0&0&0&1&0\\ 0&1&1&0&0&0&1 \end{pmatrix} \quad \text{и} \quad H=\begin{pmatrix} 1&0&0&1&1&1&0\\ 0&1&0&0&1&1&1\\ 0&0&1&1&1&0&1 \end{pmatrix}.$$

4. Метод, рассмотренный в дополнении 2, можно использовать для построения кодера систематического циклического кода.

При кодировании с помощью матрицы (4) кодовое слово получается умножением информационного слова на эту матрицу (см. (6), § 11). В этом случае последние k символов $a_m, a_{m+1}, \ldots, a_{n-1}$ кодового слова совпадают с информационными символами, а все кодовое слово является линейной комбинацией строк порождающей матрицы (4) с коэффициентами $a_m, a_{m+1}, \ldots, a_{n-1}$. Учитывая, что по строкам матрицы R

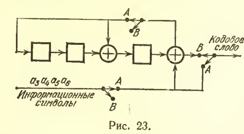


Рис. 23.

стоят коэффициенты остатков от деления многочленов x^l на порождающий многочлен $g(x)$, мы убеждаемся, что величины $a_0, a_1, \ldots, a_{m-1}$ являются коэффициентами остатка от деления многочлена

$$-(a_m x^m + a_{m+1}x^{m+1}+\ldots+a_{n-1}x^{n-1})$$

на многочлен $g(x)$ (в случае двоичных многочленов знак минус перед скобкой можно опустить).

Следовательно, схемы деления могут быть использованы для нахождения проверочных символов кодовых слов циклического кода в систематической форме и, в конечном итоге, для построения его кодера. На рис. 23 приведена схема кодера двоичного (7, 4)-кода с порождающим многочленом $1+x^2+x^3$, основанная на указанном принципе,

Проследим вкратце работу этой схемы в течение семи тактов. Первые четыре такта переключатели схемы находятся в положении A, следующие три — в положении B. В первые четыре такта информационные символы поступают (без изменений) на выход всей схемы и на выход схемы деления. В течение этих четырех тактов в последовательных ячейках схемы деления получаются коэффициенты остатка a_0, a_1, a_2 от деления многочлена $a_6x^6+a_5x^5+a_4x^4+a_3x^3$ на порождающий многочлен $1+x^2+x^3$, т. е. проверочные символы (в схеме на рис. 18 на это ушло бы семь тактов, но в данной схеме три такта «экономятся», так как последовательность коэффициентов a_6, a_5, a_4, a_3 подается не на вход, а сразу на выход схемы деления). В последующие три такта проверочные символы — сначала a_2, затем a_1, затем a_0— поступают из схемы деления на выход всей схемы. Таким образом, по истечении семи тактов на выходе всей схемы получаем целиком кодовое слово.

Рекомендуем читателю проследить по тактам работу рассматриваемой схемы для случая $a_3=0$, $a_4=a_5=a_6=1$.

17. ГОЛОСОВАНИЕ

Хотя в процедуре принятия решения большинством голосов нет для нас ничего необычного, может все же показаться неожиданным, что метод голосования используется при декодировании помехоустойчивых кодов. Отчасти мы уже коснулись этого вопроса в § 8, когда рассказывали о коде с повторением,— решение о посланном символе принималось там как раз большинством голосов. Теперь же мы покажем, как применяется голосование в случае произвольного линейного кода.

Обратимся сначала к примеру. Здесь нам поможет неоднократно упоминавшийся ранее (7,3)-код, получающийся из (7,4)-кода Хемминга добавлением общей проверки на четность. Его проверочная матрица имеет вид:

$$H=\begin{pmatrix} 1&1&1&1&1&1&1\\ 0&0&0&1&1&1&1\\ 0&1&1&0&0&1&1\\ 1&0&1&0&1&0&1 \end{pmatrix}.$$

Удобнее, однако, рассмотреть эквивалентный циклический код с порождающим многочленом $g(x)=(x+1)(x^3+x+1)$ и с проверочной матрицей

$$H_1=\begin{pmatrix} 1&1&0&1&0&0&0\\ 0&1&1&0&1&0&0\\ 0&0&1&1&0&1&0\\ 0&0&0&1&1&0&1 \end{pmatrix}.$$

Эта матрица получена способом, указанным в дополнении 9 к § 11. В том, что коды с проверочными матрицами H и H_1 действительно эквивалентны, читатель может убедиться самостоятельно.

Пусть принято некоторое слово $\alpha_0\, \alpha_1\, \alpha_2\, \alpha_3\, \alpha_4\, \alpha_5\, \alpha_6$, содержащее, быть может, ошибочные символы. Для каждого символа α_i будем в отдельности решать, верен он или нет, используя те соотношения, которыми этот символ связан с остальными. Начнем с символа α_0 и выпишем некоторые содержащие его проверки. Первой строке матрицы H_1 отвечает проверочное соотношение $\alpha_0+\alpha_1+\alpha_3=0$, сумме первой, второй и третьей строк — соотношение $\alpha_0+\alpha_4+$ $+\alpha_5=0$, а сумме первой, второй и четвертой строк — соотношение $\alpha_0+\alpha_2+\alpha_6=0$. Итак, имеем:

$$\begin{aligned} \alpha_0&=\alpha_1+\alpha_3,\\ \alpha_0&=\alpha_4+\alpha_5,\\ \alpha_0&=\alpha_2+\alpha_6. \end{aligned} \qquad (1)$$

Если ошибки при передаче отсутствовали, то в принятом слове будет выполняться каждое из соотношений (1), а правые их части дадут верное значение для α_0. Соотношения (1) для символа α_0 выбраны с таким расчетом, что всякий другой символ входит в правую часть ровно одной проверки. Поэтому если лишь один из них ошибочен, то только в одном из соотношений (1) будет нарушено равенство. Учитывая это, можно предложить следующее правило для определения верного значения символа α_0: если среди значений $\alpha_1+\alpha_3$, $\alpha_4+\alpha_5$, $\alpha_2+\alpha_6$, α_0 большинство составляют нули, то полагаем, что нуль и есть верное значение для α_0, если же большинство из них — единицы, то верным значением для α_0 считаем единицу. Такое голосование гарантирует верное решение, если принятое слово содержит не более одной ошибки. Возможно и равенство голосов (например, в случае двойной ошибки), и в этом случае приходится довольствоваться ее обнаружением.

Аналогичные проверки могут быть составлены и для других символов. Например, для α_1 имеем соотношения:

$$\begin{aligned} \alpha_1&=\alpha_2+\alpha_4,\\ \alpha_1&=\alpha_5+\alpha_6,\\ \alpha_1&=\alpha_0+\alpha_3, \end{aligned}$$

также обладающие тем свойством, что каждый символ входит в правую часть не более одного раза. Эти проверки получаются из системы (1) циклическим сдвигом на одну позицию. Последующие циклические сдвиги дают системы подобных проверок для остальных символов.

Анализируя указанным способом каждый символ принятого слова, мы правильно восстановим посланное кодовое слово, если произошло не более одной ошибки, и обна-

ружим любую двойную ошибку. Тем самым полностью используются корректирующие способности данного кода — ведь его кодовое расстояние равно 4.

Разобранный нами метод исправления и обнаружения ошибок называют *мажоритарным декодированием* (т. е. декодированием по принципу большинства). Применим он тогда, когда — как в нашем примере — для каждого символа α_j существует система проверок

$$\begin{aligned} \alpha_j &= \sum_k a_{1k}\alpha_k, \\ \alpha_j &= \sum_k a_{2k}\alpha_k, \\ &\cdots\cdots \\ \alpha_j &= \sum_k a_{sk}\alpha_k, \end{aligned} \qquad (2)$$

обладающая тем свойством, что в правую часть каждой проверки входят символы, отличные от α_j, и всякий такой символ входит не более чем в одну проверку. Такая система проверок называется *ортогональной* (или *разделенной*). Если число проверок, входящих в каждую ортогональную систему, не меньше s, то путем голосования могут быть исправлены любые t ошибок, где $t<(s+1)/2$. В самом деле, ошибка в одном символе влияет в силу ортогональности не более чем на одну проверку, следовательно, среди значений символа α_j, которые даются всеми соотношениями (2), неправильными могут оказаться не более t, т. е. не более половины значений. Тогда, сравнивая значения правых частей проверок, а также значение самого символа α_j, мы по большинству голосов определяем верное значение этого символа. Если же (при нечетном s) $t=(s+1)/2$, то имеет место равенство голосов, ошибка при этом хотя и обнаруживается, но не исправляется.

Случай, когда число s проверок в каждой ортогональной системе на единицу меньше кодового расстояния d, т. е. когда $s=d-1$, является в известном смысле идеальным. В этом случае голосование позволяет, как и в рассмотренном примере, полностью реализовать корректирующие свойства кода. Код, для каждого символа которого существует система из $(d-1)$ ортогональных проверок, называется *полностью ортогонализуемым*.

Чем хорош метод голосования? Прежде всего тем, что его техническая реализация предельно проста (особенно в случае двоичных циклических кодов). Наряду с обычными сумматорами и ячейками памяти схема декодирования дол-

жна содержать мажоритарный элемент (логическую схему, осуществляющую выбор значения α_j по большинству голосов).

Вот как устроена принципиальная схема мажоритарного декодирования для (7,3)-кода, упомянутого выше (рис. 24, *а*).

В этой схеме *М* — мажоритарный элемент, осуществляющий голосование, *К* — переключатель, находящийся в

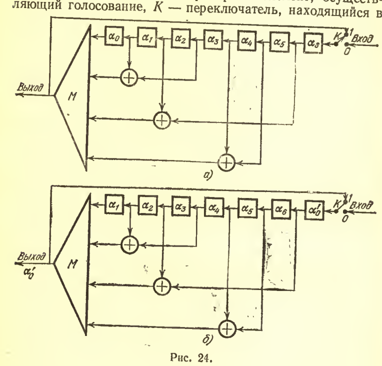


Рис. 24.

положении 0 во время приема сообщения и в положении *1* при его декодировании. Рис. 24, *а* соответствует моменту, когда сообщение полностью поступило в запоминающий регистр и ключ *К* переводится в положение *1*. Начинается декодирование. На первом его шаге (первый такт) на вход мажоритарного элемента подаются значения α_0, $\alpha_1+\alpha_3$, $\alpha_2+\alpha_6$, $\alpha_4+\alpha_5$. На выходе его появляется символ α_0', являющийся итогом голосования. В следующем такте происходит сдвиг сообщения в регистре, а символ α_0' поступает в последнюю ячейку памяти. Этому моменту соответствует рис. 24, *б*.

Теперь уже голосование осуществляется по значениям α_1, $\alpha_2+\alpha_4$, $\alpha_3+\alpha'_0$, $\alpha_5+\alpha_6$, и в результате получается значение α'_1 следующего символа сообщения. Дальнейшие такты работы схемы совершенно аналогичны. Если произошло не более одной ошибки, то последовательность $(\alpha'_0, \alpha'_1, \ldots, \alpha'_6)$ и есть верное кодовое слово.

Как мы видим, в схеме используется только информация о принятом слове; никакой дополнительной информации не требуется. А это очень важно, потому что именно необходимость хранения большого объема данных служит основным препятствием для применения некоторых методов декодирования (например, синдромного декодирования).

Еще одно немаловажное достоинство голосования по сравнению с другими методами декодирования заключается в том, что этот метод зачастую позволяет исправлять или обнаруживать многие ошибки, кратность которых превышает $(d-1)/2$.

Задачи и дополнения

1. Доказать, что код из примера § 8 с проверками на четность по строкам и по столбцам является полностью ортогонализуемым, составив для каждого из символов систему из трех ортогональных проверок (например, проверки

$$\alpha_1=\alpha_2+\alpha_3+\beta_1,$$
$$\alpha_1=\alpha_4+\alpha_7+\beta_4,$$
$$\alpha_1=\alpha_5+\alpha_6+\alpha_8+\alpha_9+\beta_3+\beta_5+\beta_6+\beta_7$$

образуют требуемую систему для символа α_1).

2. Проверить, что (7,4)-код Хемминга не является полностью ортогонализуемым, показав, что для символа α_1 не существует пары ортогональных проверок.

Будет ли полностью ортогонализуемым расширенный (8,4)-код Хемминга?

3. Решить те же вопросы, что и в задаче 2, для произвольного двоичного кода Хемминга и его расширенного варианта.

4. Построить систему ортогональных проверок и мажоритарный декодер для троичного циклического (13,6)-кода с порождающим многочленом

$$g(x)=(x+2)\,(x^3+2x^2+2x+2)\,(x^3+x^2+2).$$

18. МНОГОСТУПЕНЧАТОЕ ГОЛОСОВАНИЕ И КОДЫ РИДА — МАЛЛЕРА

Мы расскажем здесь о кодах, для которых найден замечательный алгоритм исправления ошибок, сыгравший важнейшую роль в развитии методов мажоритарного декодирования.

Проще всего начать с расширенного (8,4)-кода Хемминга. Нетрудно убедиться, что этот код дуален самому себе, т. е. что его проверочная матрица служит для него и порождающей:

$$G=\begin{pmatrix}1&1&1&1&1&1&1&1\\0&0&0&0&1&1&1&1\\0&0&1&1&0&0&1&1\\0&1&0&1&0&1&0&1\end{pmatrix}=\begin{pmatrix}g_0\\g_1\\g_2\\g_3\end{pmatrix}.$$

Этот код не является полностью ортогонализуемым (см. § 17, задача 2). Значит ли это, что голосование для этого кода невозможно? Оказывается, нет.

Идея, которую мы проиллюстрируем на данном примере, состоит в следующем. Если нельзя составить нужной системы ортогональных проверок для каждого символа в отдельности, то, может быть, это удастся сделать для определенных линейных комбинаций этих символов.

Пусть $\boldsymbol{v}=(\alpha_0, \alpha_1, \alpha_2, \alpha_3, \alpha_4, \alpha_5, \alpha_6, \alpha_7)$ — произвольное кодовое слово. Оно, как обычно, может быть записано в виде:

$$\boldsymbol{v}=a_0\boldsymbol{g}_0+a_1\boldsymbol{g}_1+a_2\boldsymbol{g}_2+a_3\boldsymbol{g}_3. \tag{1}$$

Выясним, как связаны координаты α_i вектора $\boldsymbol{v}$ с коэффициентами равенства (1). Заметим, что сумма первых двух координат каждого из векторов $\boldsymbol{g}_0$, $\boldsymbol{g}_1$ и $\boldsymbol{g}_2$ равна 0, тогда как для вектора $\boldsymbol{g}_3$ она равна 1. Поэтому сумма соответствующих координат вектора $\boldsymbol{v}$ равна a_3, т. е.

$$a_3=\alpha_0+\alpha_1. \tag{2}$$

Совершенно так же убеждаемся, что верны равенства:

$$\begin{aligned}a_3&=\alpha_2+\alpha_3,\\a_3&=\alpha_4+\alpha_5,\\a_3&=\alpha_6+\alpha_7.\end{aligned} \tag{3}$$

Нетрудно понять, что соотношения (2) и (3) представляют систему ортогональных проверок для коэффициента a_3. Для коэффициентов a_2 и a_1 дело обстоит аналогично и ортогональные проверки для них таковы:

$$\begin{aligned}a_2&=\alpha_0+\alpha_2, & a_1&=\alpha_0+\alpha_4,\\a_2&=\alpha_1+\alpha_3, & a_1&=\alpha_1+\alpha_5,\\a_2&=\alpha_4+\alpha_6, & a_1&=\alpha_2+\alpha_6,\\a_2&=\alpha_5+\alpha_7, & a_1&=\alpha_3+\alpha_7.\end{aligned} \tag{4}$$

Применяя к полученным проверкам принцип большинства, мы найдем верные значения коэффициентов a_1, a_2, a_3, если произошло не более одной ошибки. В случае двойной ошибки при определении хотя бы одного из коэффициентов голоса распределятся поровну и ошибка будет только обнаружена.

Предположим, что верные значения коэффициентов a_1, a_2, a_3 найдены. Тогда еще один этап голосования позволяет определить также и верное значение коэффициента a_0. Сделать это совсем несложно. В самом деле, рассмотрим разность

$$\boldsymbol{v}_1 = \boldsymbol{v} - a_1\boldsymbol{g}_1 - a_2\boldsymbol{g}_2 - a_3\boldsymbol{g}_3.$$

В случае безошибочной передачи, в силу равенства (1), $\boldsymbol{v}_1 = a_0\boldsymbol{g}_0$, т. е. все координаты вектора $\boldsymbol{v}_1$ равны a_0. Значит, либо все они равны нулю, либо все равны единице. Поэтому в качестве верного значения a_0 выбираем то, которое преобладает среди координат вектора $\boldsymbol{v}_1$.

Определив значения коэффициентов a_0, a_1, a_2, a_3, мы без труда восстанавливаем кодовое слово согласно равенству (1).

П р и м е р. Пусть принято слово 01110110, содержащее одиночную ошибку. Для декодирования обращаемся к равенствам (2), (3) и (4), которые в данном случае дают следующие результаты:

$$a_1=0,\quad a_2=1,\quad a_3=1,$$
$$a_1=0,\quad a_2=0,\quad a_3=0,$$
$$a_1=0,\quad a_2=1,\quad a_3=1,$$
$$a_1=1,\quad a_2=1,\quad a_3=1.$$

По большинству значений находим: $a_1=0$; $a_2=a_3=1$. Тогда

$$\boldsymbol{v}_1 = (01110110) - (00110011) - (01010101) = (00010000),$$

откуда $a_0=0$. Следовательно,

$$\boldsymbol{v} = \boldsymbol{g}_2 + \boldsymbol{g}_3 = (01100110).$$

Этот изящный метод декодирования может быть применен к так называемым кодам Рида — Маллера (сокращенно — РМ-коды), которые мы и хотим сейчас рассмотреть.

РМ-код первого порядка определяется как код, дуальный к расширенному коду Хемминга, т. е. как код, порождающая матрица которого совпадает с проверочной матрицей расширенного кода Хемминга. Таким образом, расширенному (n, k)-коду Хемминга ($n=2^m$, $k=2^m-m-1$) отвечает $(n, n-k)$-код Рида — Маллера. При $m=3$ оба кода сов-

падают: расширенный (8,4)-код Хемминга дуален самому себе. При $m=4$ получаем (16,5)-код Рида — Маллера первого порядка с порождающей матрицей

$$G=\begin{pmatrix} 1&1&1&1&1&1&1&1&1&1&1&1&1&1&1&1\\ 0&0&0&0&0&0&0&0&1&1&1&1&1&1&1&1\\ 0&0&0&0&1&1&1&1&0&0&0&0&1&1&1&1\\ 0&0&1&1&0&0&1&1&0&0&1&1&0&0&1&1\\ 0&1&0&1&0&1&0&1&0&1&0&1&0&1&0&1 \end{pmatrix}=\begin{pmatrix} g_0\\ g_1\\ g_2\\ g_3\\ g_4 \end{pmatrix}.$$

Алгоритм декодирования для этого кода практически не отличается от рассмотренного выше алгоритма для (8,4)-кода. Действительно, если

$$\boldsymbol{v}=a_0\boldsymbol{g}_0+a_1\boldsymbol{g}_1+a_2\boldsymbol{g}_2+a_3\boldsymbol{g}_3+a_4\boldsymbol{g}_4,$$

то, например,

$$a_4=\alpha_0+\alpha_1=\alpha_2+\alpha_3=\alpha_4+\alpha_5=\alpha_6+\alpha_7=\alpha_8+\alpha_9=\alpha_{10}+\alpha_{11}=$$
$$=\alpha_{12}+\alpha_{13}=\alpha_{14}+\alpha_{15}.$$

Таким образом, для коэффициента a_4 имеем 8 ортогональных проверочных соотношений. Такое же число ортогональных проверок получается и для каждого из коэффициентов a_1, a_2, a_3.

Следовательно, если произошло не более трех ошибок, то в первом «туре» голосования удается восстановить верные значения коэффициентов a_1, a_2, a_3, a_4. Во втором «туре», действуя совершенно так же, как и в случае (8,4)-кода, получаем верное значение для a_0. После этого по найденным значениям коэффициентов целиком восстанавливаем кодовое слово. Итак, (16,5)-код Рида — Маллера исправляет любые ошибки, если они имеются не более чем в трех символах, и обнаруживает ошибки в четырех символах. Можно убедиться, что двух этапов голосования достаточно для любого РМ-кода первого порядка.

При построении кодов Рида — Маллера более высоких порядков используется покомпонентное умножение векторов, определяемое следующим образом: если $\boldsymbol{v}=(\alpha_1, \alpha_2, \ldots, \alpha_n)$, $\boldsymbol{u}=(\beta_1, \beta_2, \ldots, \beta_n)$, то

$$\boldsymbol{v}\circ\boldsymbol{u}=(\alpha_1\beta_1, \alpha_2\beta_2, \ldots, \alpha_n\beta_n).$$

Покажем теперь, как по произвольному РМ-коду первого порядка строятся РМ-коды r-го порядка ($r>1$). Пусть строками порождающей матрицы РМ-кода первого порядка являются векторы $\boldsymbol{g}_0, \boldsymbol{g}_1, \ldots, \boldsymbol{g}_m$. Составим из этих векторов всевозможные произведения, содержащие не более r сомножителей. Добавим эти произведения в качестве допол-

нительных строк к строкам $g_0, g_1, \ldots, g_m$. Полученную в результате матрицу и будем считать порождающей матрицей РМ-кода r-го порядка. Например, для РМ-кода второго порядка, соответствующего (16,5)-коду Рида — Маллера первого порядка, эта матрица имеет вид

$$G=\begin{pmatrix} 1&1&1&1&1&1&1&1&1&1&1&1&1&1&1&1\\ 0&0&0&0&0&0&0&0&1&1&1&1&1&1&1&1\\ 0&0&0&0&1&1&1&1&0&0&0&0&1&1&1&1\\ 0&0&1&1&0&0&1&1&0&0&1&1&0&0&1&1\\ 0&1&0&1&0&1&0&1&0&1&0&1&0&1&0&1\\ 0&0&0&0&0&0&0&0&0&0&0&0&1&1&1&1\\ 0&0&0&0&0&0&0&0&0&0&1&1&0&0&1&1\\ 0&0&0&0&0&0&0&0&0&1&0&1&0&1&0&1\\ 0&0&0&0&0&0&1&1&0&0&0&0&0&0&1&1\\ 0&0&0&0&0&1&0&1&0&0&0&0&0&1&0&1\\ 0&0&0&1&0&0&0&1&0&0&0&1&0&0&0&1 \end{pmatrix}=\begin{pmatrix} g_0\\ g_1\\ g_2\\ g_3\\ g_4\\ g_1\circ g_2\\ g_1\circ g_3\\ g_1\circ g_4\\ g_2\circ g_3\\ g_2\circ g_4\\ g_3\circ g_4 \end{pmatrix}.$$

Этот код исправляет любые одиночные и обнаруживает двойные ошибки, а мажоритарное декодирование для него может быть осуществлено в три этапа. Если v кодовое слово, то

$$v=a_0g_0+a_1g_1+\ldots+a_4g_4+a_{12}g_1\circ g_2+ +a_{13}g_1\circ g_3+\ldots+a_{34}g_3\circ g_4. \qquad (5)$$

На первом этапе определяем верные значения коэффициентов a_{ij}, для каждого из которых имеем четыре ортогональных проверочных соотношения. Например, для коэффициента a_{34} они таковы:

$$a_{34}=\alpha_0+\alpha_1+\alpha_2+\alpha_3,$$
$$a_{34}=\alpha_4+\alpha_5+\alpha_6+\alpha_7,$$
$$a_{34}=\alpha_8+\alpha_9+\alpha_{10}+\alpha_{11},$$
$$a_{34}=\alpha_{12}+\alpha_{13}+\alpha_{14}+\alpha_{15}.$$

После определения коэффициентов a_{ij} составляем разность

$$v_1=v-a_{12}g_1\circ g_2-\ldots-a_{34}g_3\circ g_4=(\alpha_0', \alpha_1', \ldots, \alpha_{15}').$$

Поскольку при безошибочной передаче

$$v_1=a_0g_0+a_1g_1+a_2g_2+a_3g_3+a_4g_4,$$

то проверочные соотношения для определения a_1, a_2, a_3, a_4 (это второй этап голосования) таковы же, как для РМ-кода первого порядка. К примеру,

$$a_4=\alpha_0'+\alpha_1'=\alpha_2'+\alpha_3'=\alpha_4'+\alpha_5'=\alpha_6'+\alpha_7'=\alpha_8'+\alpha_9'= =\alpha_{10}'+\alpha_{11}'=\alpha_{12}'+\alpha_{13}'=\alpha_{14}'+\alpha_{15}'.$$

Наконец, на третьем этапе составляем разность

$$v_1 - a_1 g_1 - a_2 g_2 - a_3 g_3 - a_4 g_4 = (\alpha_0'', \alpha_1'', \ldots, \alpha_{15}'')$$

и по большинству значений координат α_i'' определяем значение a_0. После завершения третьего этапа кодовое слово восстанавливается по формуле (5).

Подобная процедура декодирования распространяется на РМ-коды произвольных порядков; при этом для кода r-го порядка число этапов голосования равно $r+1$ и для кода длины 2^m число ортогональных проверок на первом этапе равно 2^{m-r} (на каждом последующем этапе число проверок удваивается). Таким образом, может быть исправлено $2^{m-r-1}-1$ ошибок, а 2^{m-r-1} ошибок всегда обнаруживается.

В настоящее время открыто достаточно много кодов, для которых применимы подобные алгоритмы декодирования. Их описание и способы построения ортогональных проверок базируются на различных комбинаторных и геометрических конструкциях.

Задачи и дополнения

1. Для (8,4)-кода Рида — Маллера первого порядка исправить или обнаружить ошибки в следующих словах:

11010110, 11100001.

2. Анализируя соотношения (2), (3), (4), найти число обнаруживаемых и число необнаруживаемых тройных ошибок в (8,4)-коде Рида—Маллера. Имеются ли исправимые тройные ошибки? Решить те же вопросы для ошибок в четырех символах.

3. Для (16,5)-кода Рида — Маллера первого порядка исправить или обнаружить ошибки в следующих словах:

1110010100100111, 1010010110101010, 1001001001001001.

4. Считая, что используется РМ-код второго порядка длины 16, исправить или обнаружить ошибки в тех же словах, что и в задаче 3. Чем объясняются расхождения с результатами задачи 3?

5. Доказать, что код Рида — Маллера r-го порядка длины $n=2^m$ имеет $1+C_m^1+\ldots+C_m^r$ информационных символов, $1+C_m^1+\ldots+C_m^{m-r-1}$ проверочных символов, а его кодовое расстояние равно 2^{m-r}.

19. ЛАТИНСКИЕ КВАДРАТЫ И КОДЫ

Латинские квадраты долгое время были известны лишь математикам и любителям головоломок и, в основном, благодаря одной знаменитой задаче Л. Эйлера *). В 1782 г. Эйлер предложил следующую проблему.

*) Леонард Эйлер (1707—1783) — один из великих математиков XVIII века, создавших основы математического анализа, Швейцарец

Среди 36 офицеров находится по шесть офицеров шести различных званий из шести полков. Можно ли построить этих офицеров в каре так, чтобы в каждой колонне и каждой шеренге встречались офицеры всех званий и всех полков?

Лишь в 1901 г. удалось доказать, что это невозможно. Однако связанные с задачей Эйлера латинские квадраты не потеряли интереса, так как вскоре обнаружилось, что они имеют многообразные практические применения. А совсем недавно (конец шестидесятых годов) они были применены и в теории кодирования. Получающиеся на их основе коды, хотя и далеки по своим параметрам от оптимальных, но зато допускают простые алгоритмы декодирования (голосование в один шаг).

Будем рассматривать квадратные матрицы порядка n, элементы которых — числа 1, 2, . . ., n. Такую матрицу называют *латинским квадратом*, если всякий элемент входит ровно один раз в каждую строку и в каждый столбец.

Следующие матрицы являются примерами латинских квадратов (соответственно порядка 2, 3 и 4):

$$\begin{pmatrix} 1 & 2 \\ 2 & 1 \end{pmatrix}, \quad \begin{pmatrix} 1 & 2 & 3 \\ 2 & 3 & 1 \\ 3 & 1 & 2 \end{pmatrix}, \quad \begin{pmatrix} 1 & 2 & 3 & 4 \\ 3 & 4 & 1 & 2 \\ 2 & 3 & 4 & 1 \\ 4 & 1 & 2 & 3 \end{pmatrix}.$$

Существенным при построении кодов является свойство ортогональности матриц, которое определяется следующим образом.

Две матрицы порядка n называются *ортогональными* (не путать с ортогональностью векторов!), если при наложении любой из них на другую мы получим множество всех упорядоченных пар (i, j), $1 \leqslant i \leqslant n$, $1 \leqslant j \leqslant n$.

Вот пример ортогональных латинских квадратов порядка 3:

$$\begin{pmatrix} 1 & 2 & 3 \\ 2 & 3 & 1 \\ 3 & 1 & 2 \end{pmatrix}, \quad \begin{pmatrix} 1 & 2 & 3 \\ 3 & 1 & 2 \\ 2 & 3 & 1 \end{pmatrix}.$$

по происхождению, он жил и работал преимущественно в России. Эйлер, выделявшийся своей исключительной интуицией и разносторонностью интересов, оставил глубокий след практически во всех областях современной ему математики. Большое количество его замечательных результатов явилось основой для дальнейшего развития многих разделов математики.

Ортогональны также и следующие две матрицы:

$$A=\begin{pmatrix} 1 & 1 & 1 & \dots & 1 \\ 2 & 2 & 2 & \dots & 2 \\ 3 & 3 & 3 & \dots & 3 \\ . & . & . & . & . \\ n & n & n & \dots & n \end{pmatrix}, \quad B=\begin{pmatrix} 1 & 2 & 3 & \dots & n \\ 1 & 2 & 3 & \dots & n \\ 1 & 2 & 3 & \dots & n \\ . & . & . & . & . \\ 1 & 2 & 3 & \dots & n \end{pmatrix}. \qquad (1)$$

Легко видеть, что матрица порядка n является латинским квадратом тогда и только тогда, когда она ортогональна обеим матрицам A и B.

При построении кодов используются матрицы (1) и им ортогональные. Способ построения обеспечивает нужное для декодирования число ортогональных проверок; состоит этот способ в следующем.

Для матрицы C указанного типа и для любого ее элемента k определим двоичную матрицу C_k, которая получается из C заменой всех элементов, равных k, единицами, а всех остальных элементов — нулями. Таким образом, по матрице C порядка n строятся матрицы $C_1, C_2, \dots, C_n$. Каждой из этих матриц C_k сопоставим вектор $\boldsymbol{v}_k$, первые n координат которого являются последовательными элементами первой строки матрицы C_k, следующие n координат — элементами второй строки и т. д. Иными словами, n^2 координат вектора $\boldsymbol{v}_k$— это все элементы матрицы C_k, «вытянутой» в одну общую строку. Образуем, наконец, матрицу $\tilde{C}$ порядка $n \times n^2$, строками которой являются векторы $\boldsymbol{v}_k$:

$$\tilde{C}=\begin{pmatrix} \boldsymbol{v}_1 \\ \boldsymbol{v}_2 \\ \vdots \\ \boldsymbol{v}_n \end{pmatrix}.$$

Например, для матрицы

$$C=\begin{pmatrix} 1 & 2 & 3 \\ 2 & 3 & 1 \\ 3 & 1 & 2 \end{pmatrix}$$

имеем:

$$C_1=\begin{pmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \end{pmatrix}, \quad C_2=\begin{pmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{pmatrix}, \quad C_3=\begin{pmatrix} 0 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \end{pmatrix},$$

$$v_1=(100001010), \quad v_2=(010100001), \quad v_3=(001010100),$$

$$\tilde{C}=\begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 0 \end{pmatrix}.$$

Пусть теперь A и B — матрицы (1), а $D_1, D_2, \ldots, D_r$ — попарно ортогональные латинские квадраты. Образуем из них указанным выше способом матрицы $\tilde{A}, \tilde{B}, \tilde{D}_1, \tilde{D}_2, \ldots, \tilde{D}_r$, а затем построим матрицу H, которую можно условно представить в виде:

$$H = \begin{pmatrix} \tilde{A} & \vdots & \\ \tilde{B} & \vdots & \\ \tilde{D}_1 & \vdots & E_m \\ \tilde{D}_2 & \vdots & \\ \vdots & \vdots & \\ \tilde{D}_r & \vdots & \end{pmatrix}$$

(матрицы $\tilde{A}, \tilde{B}, \tilde{D}_1, \tilde{D}_2, \ldots, \tilde{D}_r$ подписываются одна под другой и к получившейся матрице приписывается справа единичная матрица E_m соответствующего порядка m).

Матрицу H будем считать проверочной матрицей кода, построенного с помощью латинских квадратов. Число строк этой матрицы, как вытекает из построения, равно $(r+2)n$, а число столбцов составляет $n^2+(r+2)n$. Кодовое расстояние полученного кода, как мы увидим, будет не меньше $r+3$.

В качестве примера рассмотрим код, построенный с помощью двух ортогональных латинских квадратов порядка 3:

$$D_1 = \begin{pmatrix} 1 & 2 & 3 \\ 2 & 3 & 1 \\ 3 & 1 & 2 \end{pmatrix}, \quad D_2 = \begin{pmatrix} 1 & 2 & 3 \\ 3 & 1 & 2 \\ 2 & 3 & 1 \end{pmatrix}.$$

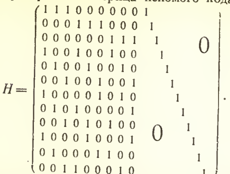
В этом случае

$$A = \begin{pmatrix} 1 & 1 & 1 \\ 2 & 2 & 2 \\ 3 & 3 & 3 \end{pmatrix}, \qquad B = \begin{pmatrix} 1 & 2 & 3 \\ 1 & 2 & 3 \\ 1 & 2 & 3 \end{pmatrix},$$

$$\tilde{A} = \begin{pmatrix} 1&1&1&0&0&0&0&0&0 \\ 0&0&0&1&1&1&0&0&0 \\ 0&0&0&0&0&0&1&1&1 \end{pmatrix}, \quad \tilde{B} = \begin{pmatrix} 1&0&0&1&0&0&1&0&0 \\ 0&1&0&0&1&0&0&1&0 \\ 0&0&1&0&0&1&0&0&1 \end{pmatrix},$$

$$\tilde{D}_1 = \begin{pmatrix} 1&0&0&0&0&1&0&1&0 \\ 0&1&0&1&0&0&0&0&1 \\ 0&0&1&0&1&0&1&0&0 \end{pmatrix}, \quad \tilde{D}_2 = \begin{pmatrix} 1&0&0&0&1&0&0&0&1 \\ 0&1&0&0&0&1&1&0&0 \\ 0&0&1&1&0&0&0&1&0 \end{pmatrix}.$$

Наконец, проверочная матрица искомого кода такова:



$$H=\left(\begin{array}{ccccccccc|cccccccccccc}
1&1&1&0&0&0&0&0&0&1&&&&&&&&&&&\\
0&0&0&1&1&1&0&0&0&&1&&&&&&&&&&\\
0&0&0&0&0&0&1&1&1&&&1&&&&&0&&&&\\
1&0&0&1&0&0&1&0&0&&&&1&&&&&&&&\\
0&1&0&0&1&0&0&1&0&&&&&1&&&&&&&\\
0&0&1&0&0&1&0&0&1&&&&&&1&&&&&&\\
1&0&0&0&0&1&0&1&0&&&&&&&1&&&&&\\
0&1&0&1&0&0&0&0&1&&&&&&&&1&&&&\\
0&0&1&0&1&0&1&0&0&&&&&&&&&1&&&\\
1&0&0&0&1&0&0&0&1&&&0&&&&&&&1&&\\
0&1&0&0&0&1&1&0&0&&&&&&&&&&&1&\\
0&0&1&1&0&0&0&1&0&&&&&&&&&&&&1
\end{array}\right).$$

Легко видеть, что для каждого из 9 информационных символов может быть составлено четыре ортогональных проверки. Например, первая, четвертая, седьмая и десятая

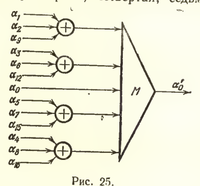


Рис. 25.

строки матрицы H дают следующие проверки для первого символа α_0:

$$\alpha_0=\alpha_1+\alpha_2+\alpha_9,$$
$$\alpha_0=\alpha_3+\alpha_6+\alpha_{12},$$
$$\alpha_0=\alpha_5+\alpha_7+\alpha_{15},$$
$$\alpha_0=\alpha_4+\alpha_8+\alpha_{18}.$$

Ортогональность этих соотношений, как можно убедиться, как раз и объясняется свойством ортогональности исходных матриц. При этом число таких соотношений для каждого символа всегда равно числу матриц, используемых для построения, т. е. $r+2$. А значит, кодовое расстояние не может быть меньше, чем $r+3$; в нашем примере оно не меньше пяти.

На рис. 25 мы приводим часть декодирующей схемы, предназначенную для декодирования символа α_0 (буквой M, как и раньше, обозначен мажоритарный элемент).

20. МАТРИЦЫ АДАМАРА И КОДИРОВАНИЕ

Важную роль в алгебре и комбинаторике играют матрицы Адамара, которые впервые были введены в математический обиход в конце прошлого века *). Сравнительно недавно (в 1960 г.) было замечено, что эти матрицы могут быть использованы для построения кодов с большим кодовым расстоянием $\left(d \geqslant \left[\frac{n}{2}\right]\right)$.

Квадратная матрица H порядка n с элементами ± 1 называется *матрицей Адамара*, если выполняется условие

$$HH^{\tau} = nE_n.$$

Вот несколько примеров матриц Адамара:

$$\underset{n=1}{(1)}, \quad \underset{n=2}{\begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}},$$

$$\underset{n=4}{\begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 \end{pmatrix}}, \quad \underset{n=8}{\begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 & 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 & 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 & 1 & -1 & -1 & 1 \\ 1 & 1 & 1 & 1 & -1 & -1 & -1 & -1 \\ 1 & -1 & 1 & -1 & -1 & 1 & -1 & 1 \\ 1 & 1 & -1 & -1 & -1 & -1 & 1 & 1 \\ 1 & -1 & -1 & 1 & -1 & 1 & 1 & -1 \end{pmatrix}}.$$

Рассмотрим произвольную матрицу Адамара

$$H = \begin{pmatrix} h_{11} & h_{12} & \dots & h_{1n} \\ h_{21} & h_{22} & \dots & h_{2n} \\ \dots & \dots & \dots & \dots \\ h_{n1} & h_{n2} & \dots & h_{nn} \end{pmatrix}, \quad h_{ij} = \pm 1.$$

Из определения следует, что для любой пары строк с номерами i и j ($i \neq j$) верно равенство:

$$h_{i1}h_{j1} + h_{i2}h_{j2} + \ldots + h_{in}h_{jn} = 0. \qquad (1)$$

*) Жак Адамар (1865—1963) — один из крупнейших французских математиков конца XIX и первой половины XX века, автор ряда основополагающих работ в области теории чисел, алгебры и математического анализа.

Таким образом, различные строки матрицы Адамара попарно ортогональны. Далее, число слагаемых в (1), равных +1, должно совпадать с числом слагаемых, равных —1. Следовательно, n четно и любые две строки совпадают ровно в $n/2$ позициях и различаются в остальных.

Пусть теперь A — двоичная матрица, получающаяся из матрицы H заменой элемента +1 на 0 —1 на 1. Множество векторов-строк матрицы A образует тогда код с расстоянием Хемминга между любыми кодовыми словами, равным $n/2$. Так, из матрицы Адамара порядка 8 получаем матрицу A, задающую код длины 8 с кодовым расстоянием 4:

$$A = \begin{pmatrix} 0&0&0&0&0&0&0&0 \\ 0&1&0&1&0&1&0&1 \\ 0&0&1&1&0&0&1&1 \\ 0&1&1&0&0&1&1&0 \\ 0&0&0&0&1&1&1&1 \\ 0&1&0&1&1&0&1&0 \\ 0&0&1&1&0&0&1&1 \\ 0&1&1&0&1&0&0&1 \end{pmatrix}. \qquad (2)$$

Не меняя кодового расстояния, можно уменьшить длину кода, если отбросить первый (нулевой) символ каждого слова.

Указанным образом из всякой матрицы Адамара порядка n можно получить двоичный код Адамара типа (n—1, n, $n/2$) (n—1 — длина кодового слова, n — число слов кода, $n/2$ — кодовое расстояние). Это, как правило, код, не являющийся линейным.

С матрицей A связаны еще два кода, которые тоже называют кодами Адамара.

Первый из них получается так. Перейдем от матрицы A к матрице $\bar{A}$, заменив все элементы матрицы A их дополнениями (т. е. заменив единицы нулями, а нули — единицами). Тогда строки матриц A и $\bar{A}$ в совокупности образуют код типа (n, $2n$, $n/2$), что легко проверяется с помощью равенства (1).

Другой код Адамара получается из предыдущего отбрасыванием первого символа в каждом кодовом слове. Это будет код типа $\left(n-1,\ 2n,\ \frac{n}{2}-1\right)$.

Например, из матрицы (2) можно получить таким способом коды Адамара типов (8,16, 4) и (7, 16, 3).

Коды Адамара, как видно из их определения, обладают интересной особенностью: расстояние между любыми двумя кодовыми словами одинаково и совпадает поэтому с кодо-

вым расстоянием. Подобные коды называют *эквидистантными*, и в некоторых случаях их использование дает особые преимущества.

Коды Адамара, обладая большим кодовым расстоянием, позволяют соответственно исправить и большое количество ошибок (первые два из них исправляют ошибки примерно в четверти позиций кодового слова). Достигается это, конечно, ценой высокой избыточности.

Для построения и реализации кода Адамара той или иной длины необходимо построить сначала матрицу Адамара соответствующего порядка. Надо сказать, что такое построение является совсем нелегким делом, и этому вопросу посвящен внушительный раздел современной комбинаторики. Некоторые методы построения будут рассмотрены ниже в разделе «Задачи и дополнения» (см. также [4]).

С матрицами Адамара связан ряд нерешенных проблем, одна из которых состоит в следующем. Мы уже видели, что порядок n матрицы Адамара при $n \geqslant 3$ может быть лишь четным. Более того, нетрудно доказать, что при $n \geqslant 4$ порядок обязан делиться на 4 (см. дополнение 3). До настоящего времени остается открытым вопрос: для любого ли n, кратного 4, существует матрица Адамара порядка n? Неизвестно, в частности, существует ли матрица Адамара порядка 268 (это наименьший порядок, кратный 4, для которого матрица Адамара еще не построена).

Задачи и дополнения

1. Укажем наиболее простой способ построения матриц Адамара сколь угодно больших порядков. Пусть H_n— матрица Адамара порядка n и $-H_n$ — матрица с противоположными элементами. Составим из них матрицу порядка $2n$ следующим образом:

$$H_{2n} = \begin{pmatrix} H_n & H_n \\ H_n & -H_n \end{pmatrix}.$$

Именно таким образом получались одна из другой матрицы порядков 1, 2, 4, 8, приведенные в начале этого параграфа.

Доказать, что матрица H_{2n} является матрицей Адамара.

2. Следующие две операции преобразуют матрицу Адамара снова в матрицу Адамара:

1) перестановка строк (или столбцов);

2) умножение строки (или столбца) на -1.

С помощью этих операций любую матрицу Адамара можно преобразовать в так называемую нормализованную матрицу Адамара, у которой первая строка и первый столбец состоят из одних единиц.

3. Докажем, что если H — матрица Адамара порядка $n>2$, то n кратно 4.

Действительно, можно считвть матрицу H нормализованной матрицей Адамара. Переставляя ее столбцы, всегда можио добиться, чтобы первые три строкн матрицы имели вид:

$$\begin{matrix} 1\,1\ldots 1 & 1 & 1\ldots & 1 & 1 & 1\ldots & 1 & 1 & 1\ldots & 1 \\ 1\,1\ldots 1 & 1 & 1\ldots & 1 & -1 & -1\ldots & -1 & -1 & -1\ldots & -1 \\ 1\,1\ldots 1 & -1 & -1\ldots & -1 & 1 & 1\ldots & 1 & -1 & -1\ldots & -1 \end{matrix}$$

Получается четыре тнпа столбцов. Пусть i, j, k, l означают соответственно чнсло столбцов первого, второго, третьего н четвертого типов. Свойство ортогоиальностн строк влечет тогда также равенства:

$$i+j-k-l=0,$$
$$i-j+k-l=0,$$
$$i-j-k+l=0.$$

Кроме того,

$$i+j+k+l=n.$$

Из этнх равенств получаем $i=j=k=l=\frac{n}{4}$, откуда н следует наше утвержденне.

4. Изложим еще однн метод построения матрицы Адамара — метод Пэлн. Рассмотрнм поле $\mathbb{Z}_p$ вычетов по модулю p, где p — простое число. Всякнй элемент $\mathbb{Z}_p$, являющнйся квадратом какого-лнбо элемента того же поля, называется *квадратичным вычетом*, всякий другой — *квадратичным невычетом*. Определим на $\mathbb{Z}_p$ следующую функцию $\chi(\bar{i})$, называемую *символом Лежандра* *):

$$\chi(\bar{i})=\begin{cases} 0, & \text{если } \bar{i}=0; \\ 1, & \text{если } \bar{i} \text{ — квадратичный вычет;} \\ -1, & \text{если } \bar{i} \text{ — квадратичный иевычет.} \end{cases}$$

Исходя нз этого определения, можно доказать, что для всякого $c\neq 0$ выполняется равенство

$$\chi(\bar{1})\,\chi(\bar{1}+\bar{c})+\chi(\bar{2})\,\chi(\bar{2}+\bar{c})+\ldots+\chi(\overline{p-1})\,\chi(\overline{p-1}+\bar{c})=-1. \quad (3)$$

Рассмотрнм теперь квадратную матрнцу Q порядка p, элементы которой q_{ij} $(i, j=1, 2, \ldots, p)$ определяются следующим образом:

$$q_{ij}=\chi(\bar{j}-\bar{i}).$$

Пусть E — едининчная матрнца порядка p, а J — квадратная матрнца того же порядка, все элементы которой равны 1. Тогда, пользуясь (3), можно доказать равенства

$$QQ^{\tau}=pE-J, \quad QJ=JQ=0. \quad (4)$$

*) Адриен Мари Лежандр (1752—1833) — французскнй математик, плодотворио работавший в теории чнсел и в ряде разделов математического анализа и механики.

Пусть теперь $p=4k-1$. В этом случае матрица

$$H=\begin{pmatrix} 1 & 1 & 1 & \dots 1 \\ 1 & \dots & \dots & \dots \\ \vdots & \vdots & & \vdots \\ 1 & \vdots & Q-E & \vdots \\ \vdots & \vdots & & \vdots \\ 1 & \dots & \dots & \dots \end{pmatrix}$$

является матрицей Адамара порядка $p+1$. Действительно, вычисляя произведение $HH^{\text{т}}$, получаем:

$$HH^{\text{т}}=\begin{pmatrix} p+1 & 0 \ \dots \ 0 \\ 0 & \\ \vdots & J+(Q-E)\,(Q^{\text{т}}-E) \\ 0 & \end{pmatrix}.$$

Далее, как нетрудно проверить, матрица Q порядка $p=4k-1$ совпадает с матрицей $-Q^{\text{т}}$. Отсюда с учетом (4) имеем:

$$J+(Q-E)\,(Q^{\text{т}}-E)=J+QQ^{\text{т}}-Q-Q^{\text{т}}+E=$$
$$=J+pE-J-Q+Q+E=(p+1)\,E.$$

Таким образом, $HH^{\text{т}}=(p+1)E$.

5. В качестве примера построим матрицу Адамара порядка 8. При этом $p=7$. Функция $\chi(i)$ задается следующей таблицей:

$\bar{i}$	$\bar{0}$	$\bar{1}$	$\bar{2}$	$\bar{3}$	$\bar{4}$	$\bar{5}$	$\bar{6}$
$\chi(\bar{i})$	0	1	1	−1	1	−1	−1

Матрицы Q и H имеют тогда вид:

$$Q=\begin{pmatrix} 0 & 1 & 1 & -1 & 1 & -1 & -1 \\ -1 & 0 & 1 & 1 & -1 & 1 & -1 \\ -1 & -1 & 0 & 1 & 1 & -1 & 1 \\ 1 & -1 & -1 & 0 & 1 & 1 & -1 \\ -1 & 1 & -1 & -1 & 0 & 1 & 1 \\ 1 & -1 & 1 & -1 & 1 & 0 & 1 \\ 1 & 1 & -1 & 1 & -1 & -1 & 0 \end{pmatrix},$$

$$H=\begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & 1 & -1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 & 1 & -1 & 1 & -1 \\ 1 & -1 & -1 & -1 & 1 & 1 & -1 & 1 \\ 1 & 1 & -1 & -1 & -1 & 1 & 1 & -1 \\ 1 & -1 & 1 & -1 & -1 & -1 & 1 & 1 \\ 1 & 1 & -1 & 1 & -1 & -1 & -1 & 1 \\ 1 & 1 & 1 & -1 & 1 & -1 & -1 & -1 \end{pmatrix}$$

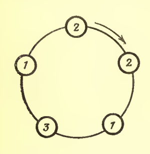
6. Построить методом Пэли матрицу Адамара порядка 12 и найти соответствующие коды Адамара.

21. ЗАДАЧА ОБ ОЖЕРЕЛЬЯХ, ФУНКЦИЯ МЁБИУСА И СИНХРОНИЗИРУЕМЫЕ КОДЫ

Некоторые вопросы теории кодирования связаны с известной комбинаторной задачей, называемой «задачей об ожерельях». Эту задачу можно сформулировать следующим образом.

Пусть ожерелье состоит из бусинок нескольких цветов. Спрашивается, сколько существует различных ожерелий, если задано число бусинок каждого цвета.

Расположим n бусинок по окружности, указав для каждой бусинки номер ее цвета. Если обойти эту окружность в определенном направлении, начав с некоторой бусинки, то ожерелью из n бусинок будет сопоставлено слово $(a_1\ a_2 \ldots a_n)$, где a_i есть номер цвета i-й бусинки. Но обход можно начать с любой бусинки, поэтому любому ожерелью соответствует n слов, получаемых одно из другого циклическими сдвигами. Например, для ожерелья, представленного на рисунке, получаем следующие слова:



12213, 31221, 13122,
21312, 22131.

Вообще говоря, среди n слов, сопоставленных ожерелью, могут быть и одинаковые. Нас, однако, будут интересовать не все ожерелья, а только такие, которые нельзя составить из двух или более одинаковых «кусков». Таким ожерельям отвечают слова, которые нельзя составить из нескольких одинаковых слов меньшей длины. Будем называть подобные слова *основными*. Например, слова 11, 100100100 не являются основными, а слово 1011 — основное. Ясно, что каждому «несоставному» ожерелью отвечает n различных основных слов.

Чтобы указать удобную формулу для числа основных слов, нам потребуется так называемая функция Мёбиуса *),

*) Август Фердинанд Мёбиус (1790—1868) — немецкий математик и астроном, известный, главным образом, своими работами по проективной геометрии. Им были впервые систематически изучены свойства функции $\mu(n)$.

определяемая следующим образом:

$$\mu(n)=\begin{cases} 1, & \text{если } n=1; \\ (-1)^k, & \text{если } n \text{ — произведение } k \text{ различных простых чисел;} \\ 0 & \text{в остальных случаях.} \end{cases}$$

Обозначим через $P_n(m)$ общее число основных слов длины n алфавита из m символов. Тогда, как можно доказать,

$$P_n(m)=\mu(d_1)\,m^{n/d_1}+\mu(d_2)\,m^{n/d_2}+\ldots+\mu(d_k)\,m^{n/d_k}, \quad (1)$$

где $d_1, d_2, \ldots, d_k$— все различные делители числа n.

Формула (1) позволяет найти число интересующих нас ожерелий, которое, очевидно, равно $P_n(m)/n$.

Выясним теперь, какое отношение имеет задача об ожерельях к проблемам кодирования. Дело в том, что при передаче закодированных сообщений должна соблюдаться определенная синхронность работы на передающей и приемной сторонах канала связи, которая обеспечивается дополнительными устройствами — тактовыми генераторами. При сбоях этих генераторов происходит нарушение синхронизации, и в качестве начального символа кодового слова воспринимается символ, который начальным не является. Отсюда актуальность задачи построения кодов, способных восстанавливать синхронизацию. Один из возможных путей ее решения (если отвлечься от исправления ошибок в символах) состоит в следующем. Будем рассматривать множество n-буквенных кодовых слов, удовлетворяющее такому ограничению: если $(a_1\, a_2 \ldots a_n)$ и $(b_1\, b_2 \ldots b_n)$ — кодовые слова (не обязательно различные), то никакое из «перекрытий» между ними

$$(a_2a_3\ldots a_nb_1),\ (a_3\ldots a_nb_1b_2),\ \ldots,\ (a_nb_1\ldots b_{n-1})$$

не является кодовым словом. Коды с этим свойством называют синхронизируемыми, и они, как легко понять, отвечают поставленной цели.

Как и обычно, платой за усовершенствование кода является уменьшение числа кодовых слов и, как обычно, возникает вопрос, насколько велика эта плата. Для ответа на этот вопрос как раз и можно использовать решение задачи об ожерельях. В самом деле, если $a=(a_1\, a_2\, a_3 \ldots a_n)$ — кодовое слово синхронизируемого кода, то кодовым словом не может быть никакой его циклический сдвиг, так как он является перекрытием для пары (a, a). Кроме того, по той же причине всякое кодовое слово должно быть основным.

Поэтому максимальное число n-буквенных слов синхронизируемого кода, использующего алфавит из m символов, не превосходит числа несоставных ожерелий с n бусинками m различных цветов. Обозначив это число через $W_n(m)$, имеем, следовательно,

$$W_n(m) \leqslant \frac{1}{n} P_n(m).$$

Таким образом, пользуясь (1), получаем такую верхнюю границу для числа n-буквенных кодовых слов синхронизируемого кода:

$$W_n(m) \leqslant \frac{1}{n} (\mu(d_1) m^{n/d_1} + \ldots + \mu(d_k) m^{n/d_k}); \qquad (2)$$

(здесь $d_1, \ldots, d_k$ по-прежнему все различные делители n).

Можно доказать (но это достаточно сложно), что при нечетных n граница (2) действительно достигается, и что это, вообще говоря, не так, если n четно. Относительно простые выкладки показывают, что при больших n сумма в скобках близка к m^n, т. е. к общему числу слов длины n в m-буквенном алфавите, так что число допустимых слов синхронизируемого кода примерно в n раз меньше общего числа слов длины n.

Задачи и дополнения

1. Доказать следующее свойство функции Мёбиуса: $\mu(nm) = \mu(n)\mu(m)$ для любых взаимно простых n и m.

2. Функция f называется *сумматорной* функцией для g, если

$$f(n) = \sum_{d|n} g(d)$$

(запись $d|n$ означает, что суммирование распространяется на все различные делители числа n).

Показать, что сумматорная функция для функции Мёбиуса равна нулю при $n>1$ и единице при $n=1$.

3. Пусть f — сумматорная функция для g (см. задачу 2). Верна следующая замечательная *формула обращения* Мёбиуса, лежащая в основе всех приложений функции $\mu(n)$:

$$g(n) = \sum_{d|n} \mu(n/d) f(d). \qquad (3)$$

Действительно,

$$\sum_{d|n} \mu(n/d) f(d) = \sum_{d|n} \mu(n/d) \sum_{k|d} g(k) = \sum_{d|n} \sum_{k|d} \mu(n/d) g(k). \qquad (4)$$

В получившейся двойной сумме изменим порядок суммирования. Заметим для этого, что аргумент k функции $g(k)$ при двойном суммирова-

нии пробегает всевозможные делители числа n, и если k фиксировано, то d пробегает все делители n, которые в свою очередь делятся на k. Поэтому двойную сумму (4) можно переписать в виде:

$$\sum_{k|n}\sum_{k|d|n} g(k)\,\mu(n/d)=\sum_{k|n} g(k)\sum_{k|d|n}\mu(n/d), \qquad (5)$$

при этом запись $k|d|n$ обозначает, что d пробегает все делители n, делящиеся на k. Введем обозначения $m=n/d$ и $t=n/k$, тогда m пробегает всевозможные делители t и в силу утверждения задачи 2

$$\sum_{k|d|n}\mu(n/d)=\sum_{m|t}\mu(m)=\begin{cases}1, & \text{если } t=1,\\ 0, & \text{если } t>1.\end{cases}$$

Следовательно,

$$\sum_{k|d|n}\mu(n/d)=\begin{cases}1, & \text{если } k=n,\\ 0, & \text{если } k<n\end{cases} \quad (k\text{ — делитель } n).$$

Обращаясь теперь к выражению (5), мы видим, что в сумме по k имеется лишь одно ненулевое слагаемое, отвечающее значению $k=n$, и потому

$$\sum_{k|n} g(k)\sum_{k|d|n}\mu(n/d)=g(n),$$

что равносильно (3).

4. Назовем число d периодом слова a, если a получается сочленением одинаковых слов длины d и не является сочленением одинаковых слов меньшей длины. Пусть $P_d(m)$ означает общее число слов длины n в алфавите из m символов, имеющих период d.

Убедиться, что

$$\sum_{d|n} P_d(m)=m^n,$$

и, пользуясь формулой обращения (3), получить оценку (2).

ЗАКЛЮЧЕНИЕ

Наши рассказы о кодировании подошли к концу. Мы познакомились, конечно, далеко не со всеми задачами этой теории, но в той или иной мере затронули три ее основных направления — кодирование с целью засекречивания сообщений, кодирование с целью сжатия информации, наконец, кодирование с целью исправления и обнаружения ошибок. Нам хотелось дать представление читателю, сколь широки и многообразны связи этой теории с разными разделами математики, и как порой самые абстрактные математические теории могут с успехом применяться для решения практических задач.

Дальнейшее проникновение в проблематику теории кодирования (подробное изучение циклических кодов и их обобщений, знакомство с кодами, исправляющими пакеты ошибок, и специфическими алгоритмами их декодирования и т. д.) требует уже значительно более серьезной математической подготовки. Впрочем, интересующийся читатель может при желании обратиться к литературе, список которой приводится в конце книги.

ПРИЛОЖЕНИЕ

В школьной математике рассматриваются различные операции над числами. Простейшие из них — сложение и обратная к нему операция вычитания, и умножение, для которого обратной операцией является деление. Подобные действия приходится производить не только над числами, но и над другими, более сложными объектами. Это можно обнаружить уже и в школьном курсе: на уроках геометрии и физики учат складывать и вычитать векторы, а на уроках алгебры рассматривают операции сложения и умножения многочленов. Однако список таких объектов гораздо обширнее. Мы познакомимся с некоторыми из них, наиболее важными для наших целей.

1. Сравнения и классы вычетов

Исходным материалом для нас остаются пока все же числа. Два целых числа a и b называют *сравнимыми по модулю n* (n — натуральное), если их разность $a-b$ делится на n без остатка *). Это выражают следующей записью:

$$a \equiv b \pmod{n}. \qquad (1)$$

Число n называют модулем сравнения (1). Например, $35 \equiv 2 \pmod{11}$, так как разность $35-2=33$ делится на 11; аналогично, $25 \equiv -11 \pmod{9}$, так как $25-(-11)=36$ делится на 9.

Запись $a \equiv 0 \pmod{n}$ означает тогда, что само число a делится на n, т. е. $a=k \cdot n$.

Если зафиксировать некоторый модуль сравнения n, то всякое натуральное число c можно единственным образом представить в виде

$$c=kn+r, \qquad (2)$$

где k — частное от деления на n, а r — остаток, совпадающий с одним

*) Понятие сравнимости впервые было введено великим немецким математиком Карлом Фридрихом Гауссом (1777—1855) в его трактате «Арифметические исследования» и является одним из основных понятий теории чисел.

из чисел

$$0,\ 1,\ 2,\ \ldots,\ n-1. \qquad (3)$$

Остаток r называют также *вычетом* числа c по модулю n. Заметим, что запись вида (2), где $0 \leqslant r \leqslant n-1$, допускают не только натуральные, но и любые целые числа. Очевидно, из равенства (2) следует, что $c \equiv r (\mathrm{mod}\ n)$, т. е. всякое число сравнимо со своим остатком (вычетом) по модулю n. Пусть теперь a и b — два произвольных числа, записанные в виде (2):

$$a = k_1 n + r_1, \quad b = k_2 n + r_2.$$

Каждый из остатков r_1 и r_2 — это одно из чисел (3), поэтому их разность может делиться на n лишь в одном случае, когда $r_1 = r_2$. Но тогда и разность $a - b = (k_1 - k_2) n + r_1 - r_2$ может делиться на n тогда и только тогда, когда $r_1 = r_2$. Отсюда следует, что $a \equiv b \pmod n$ тогда и только тогда, когда числа a и b имеют одинаковые остатки при делении на n.

В теории чисел (см., например, [5]) доказывается ряд свойств сравнений, во многом аналогичных свойствам обычных равенств. Подобно тому, как мы это делаем с равенствами, сравнения по одинаковому модулю можно складывать, перемножать и т. д. (так, перемножив сравнения $17 \equiv 5 \pmod 4$ и $7 \equiv 3 \pmod 4$, получим, как нетрудно убедиться, верное сравнение $119 \equiv 15 \pmod 4$). Вообще, если $a_1 \equiv b_1$, $a_2 \equiv b_2$, то $a_1 + a_2 \equiv b_1 + b_2$, $a_1 a_2 \equiv b_1 b_2$.

Значение этих свойств заключается в том, что при рассмотрении вопросов делимости чисел и различных числовых арифметических выражений мы можем входящие в эти выражения числа заменять на другие, сравнимые с ними по данному модулю n; в частности, каждое число может быть заменено своим вычетом. Проиллюстрируем сказанное следующей задачей.

Доказать, что число $(1981)^k + (1982)^k$ при любом нечетном натуральном k делится на 3.

Замечаем, что $1981 \equiv 1 \pmod 3$, $1982 \equiv 2 \pmod 3$. Заменяя в исходном выражении числа 1981, 1982 их вычетами по модулю 3, получаем

$$(1981)^k + (1982)^k \equiv 1 + 2^k \pmod 3.$$

Следовательно, ле́вая часть сравнения делится на 3 тогда и только тогда, когда на 3 делится сумма $1 + 2^k$. Для степеней двойки имеем: $2^2 \equiv 1$, $2^3 \equiv 2$, $2^4 \equiv 1$, $2^5 \equiv 2$ и т. д. Вообще, применяя индукцию по k, убеждаемся, что $2^k \equiv 1$ при k четном и $2^k \equiv 2$ при k нечетном. Таким образом, при нечетном k

$$1 + 2^k \equiv 1 + 2 \equiv 0 \pmod 3,$$

т. е. если k нечетно, то исходное выражение делится на 3.

В разобранной задаче числа 1981 и 1982 могли быть заменены любыми числами a и b, дающими при делении на 3 остатки соответственно 1 и 2. Ни утверждение задачи, ни способ его доказательства от этого не изменились бы. Таким образом, в некоторых вопросах все числа, имеющие один и тот же вычет r по модулю n, и, следовательно, сравнимые между собой по этому модулю, оказываются взаимозаменяемыми. Объединим все их в один класс, обозначаемый $\overline{r}$:

$$\overline{r} = \{c \mid c \equiv r \pmod{n}\}. \qquad (4)$$

Иными словами, класс $\overline{r}$ состоит из всех тех целых чисел, которые записываются в виде (2). Класс, определяемый равенством (4), называют *классом вычетов*. Каждому вычету 0, 1, 2, ..., $n-1$ отвечает свой класс вычетов, так что имеется ровно n различных классов

$$\overline{0},\quad \overline{1},\quad \overline{2},\quad \ldots,\quad \overline{n-1}. \qquad (5)$$

Ясно, что эти классы попарно не пересекаются и каждое целое число попадает ровно в один класс.

Мы обнаруживаем, далее, что используя операции сложения и умножения чисел, можно производить аналогичные операции и над классами вычетов.

В самом деле, пусть $\overline{r_1}$ и $\overline{r_2}$ — два класса вычетов. Выберем любые два числа из этих классов: $a_1 \in \overline{r_1}$, $a_2 \in \overline{r_2}$. Пусть оказалось, что сумма $a_1 + a_2$ имеет вычет r, а произведение $a_1 a_2$ — вычет s:

$$a_1 + a_2 \in \overline{r}, \qquad a_1 a_2 \in \overline{s}.$$

Тогда будем считать, что «сумма» классов $\overline{r_1}$ и $\overline{r_2}$ равна $\overline{r}$, а их «произведение» равно $\overline{s}$:

$$\overline{r_1} + \overline{r_2} = \overline{r}, \qquad \overline{r_1} \cdot \overline{r_2} = \overline{s}.$$

Законность этого определения обосновывается тем, что класс, которому принадлежит сумма $a_1 + a_2$ (соответственно произведение $a_1 a_2$) не зависит от выбора элементов a_1 и a_2 в классах $\overline{r_1}$ и $\overline{r_2}$.

Поясним данное определение на примере, взяв за модуль сравнения число $n=2$. В этом случае имеем два класса вычетов — $\overline{0}$ и $\overline{1}$, а операции над ними выглядят так:

$$\overline{0}+\overline{0}=\overline{0}; \quad \overline{0}+\overline{1}=\overline{1}+\overline{0}=\overline{1}; \quad \overline{1}+\overline{1}=\overline{0};$$
$$\overline{0}\cdot\overline{0}=\overline{0}; \quad \overline{0}\cdot\overline{1}=\overline{1}\cdot\overline{0}=\overline{0}; \quad \overline{1}\cdot\overline{1}=\overline{1}.$$

Часто, когда это не вызывает путаницы, в обозначениях классов вычетов опускают черту, записывая их как обычные натуральные числа. В основном тексте книги это делается без специальных оговорок. Выпишем, например, таблицы сложения и умножения классов по

модулю 4 в этих упрощенных обозначениях:

+	0	1	2	3
0	0	1	2	3
1	1	2	3	0
2	2	3	0	1
3	3	0	1	2

·	0	1	2	3
0	0	0	0	0
1	0	1	2	3
2	0	2	0	2
3	0	3	2	1

Эти таблицы можно понимать и буквально, считая, что они определяют две операции на множестве $\{0, 1, 2, 3\}$ — сложение и умножение по модулю 4.

2. Группы

Прежде чем дать определение группы, рассмотрим один важный пример.

Будем исходить из понятия *взаимно однозначного отображения* множества на себя. В случае конечного множества такое отображение называют обычно *подстановкой*. Пусть множество M состоит из чисел 1, 2, 3: $M=\{1, 2, 3\}$. Чтобы задать какую-либо подстановку множества M, достаточно указать для каждого из чисел 1, 2, 3 то число, в которое оно отображается этой подстановкой. Удобнее всего сделать это, использовав таблицу из двух строк,— в первой строке выписываются (в любом порядке) числа 1, 2, 3, а во второй под каждым из них пишется соответствующее ему при данной подстановке число. Скажем, обе таблицы

$$\begin{pmatrix}1 & 2 & 3\\ 2 & 3 & 1\end{pmatrix}, \quad \begin{pmatrix}2 & 3 & 1\\ 3 & 1 & 2\end{pmatrix}$$

задают одну и ту же подстановку: число 1 переходит в число 2, 2 — в 3, 3 — в 1. Нетрудно указать все различные подстановки множества M; их будет шесть:

$$\sigma_1=\begin{pmatrix}1 & 2 & 3\\ 1 & 2 & 3\end{pmatrix}, \quad \sigma_2=\begin{pmatrix}1 & 2 & 3\\ 1 & 3 & 2\end{pmatrix}, \quad \sigma_3=\begin{pmatrix}1 & 2 & 3\\ 2 & 1 & 3\end{pmatrix},$$

$$\sigma_4=\begin{pmatrix}1 & 2 & 3\\ 2 & 3 & 1\end{pmatrix}, \quad \sigma_5=\begin{pmatrix}1 & 2 & 3\\ 3 & 1 & 2\end{pmatrix}, \quad \sigma_6=\begin{pmatrix}1 & 2 & 3\\ 3 & 2 & 1\end{pmatrix}.$$

Определим теперь на множестве $S=\{\sigma_1, \sigma_2, \ldots, \sigma_6\}$ всех подстановок операцию умножения (или *композицию*) подстановок. Именно, произведением подстановок σ_i и σ_j будем называть подстановку σ_k, получающуюся в результате последовательного выполнения сначала подстановки σ_i, а затем σ_j (записывается: $\sigma_i\sigma_j=\sigma_k$). Например,

$$\begin{pmatrix}1 & 2 & 3\\ 3 & 1 & 2\end{pmatrix}\begin{pmatrix}1 & 2 & 3\\ 1 & 3 & 2\end{pmatrix}=\begin{pmatrix}1 & 2 & 3\\ 2 & 1 & 3\end{pmatrix}$$

(первая подстановка переводит число 1 в число 3, после чего вторая подстановка переводит число 2 в число 3, так что в результате последовательного выполнения подстановок число 1 перейдет в число 2 и т. д.). Читатель может проверить, что

$$\begin{pmatrix}1 & 2 & 3\\ 1 & 3 & 2\end{pmatrix}\begin{pmatrix}1 & 2 & 3\\ 3 & 1 & 2\end{pmatrix}=\begin{pmatrix}1 & 2 & 3\\ 3 & 2 & 1\end{pmatrix},$$

и тем самым убедиться, что произведение двух подстановок зависит, вообще говоря, от порядка сомножителей.

Множество S с определенной выше операцией умножения называют *группой подстановок* множества M.

Операция умножения на S обладает следующими свойствами. Во-первых, эта операция ассоциативна: для любых σ_i, σ_j, σ_k

$$(\sigma_i\sigma_j)\,\sigma_k=\sigma_i\,(\sigma_j\sigma_k).$$

Во-вторых, тождественная подстановка

$$\sigma_1=\begin{pmatrix}1 & 2 & 3\\ 1 & 2 & 3\end{pmatrix},$$

переводящая каждый элемент в себя, играет роль единицы в обычном умножении, а именно, для любой подстановки σ_i

$$\sigma_1\sigma_i=\sigma_i\sigma_1=\sigma_i.$$

Наконец, вместе с каждой подстановкой σ_i множество S содержит обратную ей подстановку σ_j, которая переводит число a в число b тогда и только тогда, когда σ_i переводит число b в число a. При этом, понятно, каждое из произведений $\sigma_i\sigma_j$ и $\sigma_j\sigma_i$ будет тождественной подстановкой:

$$\sigma_i\sigma_j=\sigma_j\sigma_i=\sigma_1.$$

Эти три свойства сохранятся, если распространить определение умножения подстановок множества $\{1, 2, 3\}$ на случай подстановок любого конечного множества.

Более того, выяснилось, что имеются самые разнообразные системы объектов с операцией, обладающей указанными свойствами.

Это привело к следующему (общему) определению группы *).

Множество G, на котором определено умножение элементов, называется *группой*, если выполняются три аксиомы:

1. Умножение ассоциативно: для любых элементов a, b, c из G

$$a(bc) = (ab)c.$$

2. В множестве G существует такой *единичный* элемент e, что

$$ea = ae = a.$$

3. Для каждого элемента a множества G существует в G такой *обратный* элемент a^{-1}, что

$$aa^{-1} = a^{-1}a = e.$$

Обращаем внимание читателя на то, что в приведенном определении говорится об умножении элементов и используется связанная с обычным умножением терминология и символика. Этой «мультипликативной» терминологии мы и будем в дальнейшем придерживаться. Однако термин «умножение» нельзя понимать буквально; речь идет о любой операции, удовлетворяющей данным аксиомам. В частности, в ряде случаев естественнее называть операцию сложением и соответственно переформулировать аксиомы группы, пользуясь терминологией, связанной со сложением («аддитивной» терминологией):

1. Сложение ассоциативно: для любых элементов a, b, c из G

$$(a+b)+c = a+(b+c).$$

2. Существует такой *нулевой* элемент 0, что для любого элемента a из G

$$a+0 = 0+a = a.$$

3. Для каждого элемента a существует такой *противоположный* элемент $(-a)$, что

$$a+(-a) = (-a)+a = 0.$$

Группа подстановок, определенная выше, является, как мы видим, одним из примеров группы. Вообще взаимно однозначные преобразования любого множества образуют группу относительно операции умножения преобразований (эти группы сыграли основную роль в формировании общего понятия группы).

Читатель обнаружит далее, что многие примеры групп ему уже известны. Перечислим некоторые из них:

---

*) В виде, близком к современному, определение группы было впервые сформулировано английским математиком А. Кэли в 1854 году.

1. Множество всех ненулевых действительных чисел относительно операции умножения — «мультипликативная группа действительных чисел» $\mathbb{R}^*$.

2. Множество всех целых чисел относительно операции сложения — «аддитивная группа целых чисел» $\mathbb{Z}$.

3. Множество всех векторов в пространстве относительно операции сложения векторов.

4. Множество всех многочленов с действительными коэффициентами относительно операции сложения многочленов.

Группы в примерах 1—4 имеют бесконечно много элементов, а группа S — конечная (6 элементов). Число элементов конечной группы называется ее *порядком*.

Из аксиом группы можно вывести, что в группе существует только один единичный элемент. Можно доказать также и единственность обратного элемента для всякого элемента группы.

Пример группы подстановок показывает, что умножение в группе может быть некоммутативным. Те группы, для которых операция коммутативна, так и называются *коммутативными* (примеры 1—4).

Если некоторое подмножество H группы G само образует группу относительно операции, определенной в G, то подмножество H называют *подгруппой* группы G.

Например, подмножества $H_1=\{\sigma_1, \sigma_2\}$ и $H_2=\{\sigma_1, \sigma_4, \sigma_5\}$ группы S являются подгруппами этой группы. Подмножество четных целых чисел есть подгруппа аддитивной группы $\mathbb{Z}$ всех целых чисел, а подмножество нечетных чисел не будет подгруппой этой группы (сложение на $\mathbb{Z}$ не задает операцию на этом подмножестве, так как сумма двух нечетных чисел есть число четное).

Имеет место следующая теорема.

Для того чтобы подмножество H являлось подгруппой группы G, необходимо и достаточно, чтобы выполнялись два условия:

1) *произведение любых элементов h_1, h_2 из подмножества H также является элементом из H ($h_1h_2 \in H$);*

2) *если $h \in H$, то и обратный к нему элемент принадлежит H ($h^{-1} \in H$).*

Наиболее просты так называемые *циклические подгруппы*, которыми обладает любая группа G. Со всяким элементом $g \in G$ можно связать «порожденную» им циклическую подгруппу, которая, по существу, представляет собой наименьшую из подгрупп, содержащую данный элемент. Чтобы определить ее, введем понятие степени элемента g, полагая $g^0=e$ и для любого натурального k

$$g^k = \underbrace{gg\ldots g}_{k \text{ сомножителей}}, \quad g^{-k}=(g^k)^{-1}.$$

При таком определении степени выполняются хорошо знакомые

нам правила действий со степенями: для любых целых чисел m и n

$$g^m g^n = g^{m+n}, \quad (g^m)^n = g^{mn}.$$

Ясно, что всякая подгруппа, содержащая элемент g, должна содержать любые его степени. С другой стороны, уже само множество степеней образует, как нетрудно убедиться, группу.

Указанную группу называют циклической подгруппой, порожденной элементом g; ее обозначают символом (g), а сам элемент g называется *образующим элементом* этой группы.

Может случиться, что для некоторого элемента g циклическая подгруппа (g) совпадает со всей группой G, и тогда группа G также называется циклической. Например, аддитивная группа целых чисел есть циклическая группа с образующим элементом 1 (или -1), а ее подгруппа четных чисел — циклическая с образующим элементом 2. Являясь наиболее простыми, циклические группы служат важным инструментом для изучения групп, имеющих более сложное строение.

Между группой и любой ее подгруппой существует определенная связь, которую можно охарактеризовать с помощью понятия смежного класса по подгруппе.

Пусть H — подгруппа группы G и $g \in G$ — произвольный элемент группы G (мы не требуем, чтобы он обязательно принадлежал подгруппе). Множество всевозможных произведений gh, где h — любой элемент подгруппы H, называется *смежным классом* (точнее, *левым смежным классом*) по подгруппе H, а элемент g — его *представителем*. Обозначая левый смежный класс через gH, имеем, следовательно,

$$gH = \{gh \mid h \in H\}.$$

Аналогично определяются правые смежные классы Hg. Если группа G — коммутативная, то $Hg = gH$.

Для смежных классов, безразлично, левых или правых, справедливы два основных факта.

1. В случае конечной подгруппы H число элементов в любом смежном классе одно и то же и совпадает с порядком подгруппы.

2. Любые два смежных класса g_1H и g_2H либо совпадают, либо вовсе не имеют общих элементов.

Для иллюстрации понятия смежного класса рассмотрим следующий пример. Пусть $\mathbb{Z}$ — аддитивная группа целых чисел, а H — ее подгруппа, состоящая из чисел, кратных пяти. Тогда можно «выписать» следующие смежные классы:

$$H+0 = \{\ldots, -10, -5, 0, 5, 10, \ldots\},$$
$$H+1 = \{\ldots, -9, -4, 1, 6, 11, \ldots\},$$
$$H+2 = \{\ldots, -8, -3, 2, 7, 12, \ldots\},$$
$$H+3 = \{\ldots, -7, -2, 3, 8, 13, \ldots\},$$
$$H+4 = \{\ldots, -6, -1, 4, 3, 14, \ldots\}.$$

Читатель сразу же заметит, что это классы вычетов $\bar{0}$, $\bar{1}$, $\bar{2}$, $\bar{3}$, $\bar{4}$ по модулю 5. Легко видеть, что всякий смежный класс $H+n$ совпадает с одним из этих пяти, и что классы эти не пересекаются. Непосредственно видно также, что

$$\mathbb{Z}=(H+0)\cup(H+1)\cup(H+2)\cup(H+3)\cup(H+4).$$

Оказывается, то же самое верно для любой группы: если Hg_1, Hg_2, ..., Hg_r— все различные (правые) смежные классы группы G по подгруппе H, то

$$G=Hg_1\cup Hg_2\cup\ldots\cup Hg_r.$$

В случае, когда G и H — конечные группы порядков m и n, из последнего равенства вытекает, что $m=n\cdot r$. Мы пришли к следующей теореме Лагранжа *):

Порядок любой подгруппы конечной группы является делителем порядка группы.

Мы привели здесь простейшие определения и факты теории групп. За дальнейшими сведениями читатель может обратиться, например, к книгам [8], [9].

3. Кольца и поля

Часто приходится рассматривать множества, на которых определены две операции, например, различные числовые множества, множества многочленов, множества классов вычетов с определенными на них операциями сложения и умножения. Выделяя некоторые свойства этих операций, общие для всех указанных множеств, мы приходим к еще одному важному алгебраическому понятию — понятию кольца.

Кольцом называется (непустое) множество K, на котором определены две операции (сложение и умножение), обладающие следующими свойствами:

1) множество K относительно сложения образует коммутативную группу;

2) умножение ассоциативно: для любых $a, b, c\in K$

$$(ab)c=a(bc);$$

*) Лагранж (1736—1813) — великий французский математик; его труды по математическому анализу, механике и теории чисел имеют важнейшее значение для развития этих дисциплин. Лагранж — один из создателей дифференциального исчисления, классической теории дифференциальных уравнений и вариационного исчисления.

3) сложение и умножение подчиняются дистрибутивному закону:

$$(a+b)c=ac+bc, \quad c(a+b)=ca+cb$$

для любых a, b, $c \in K$.

При этом множество K, рассматриваемое лишь относительно операции сложения, называется *аддитивной группой* кольца.

Приведем некоторые примеры колец.

1. Множество целых чисел с операциями сложения и умножения — кольцо целых чисел $\mathbb{Z}$.

2. Множество многочленов от одного неизвестного с действительными коэффициентами с операциями сложения и умножения многочленов — кольцо многочленов $\mathbb{R}[X]$.

3. Множество классов вычетов по модулю n с операциями сложения и умножения классов — кольцо классов вычетов $\mathbb{Z}_n$.

Читателю предлагается проверить выполнимость аксиом кольца в каждом из примеров. Остановимся подробнее на примере 3.

Поскольку операции над классами вычетов сводятся к операциям над числами из этих классов, то свойства ассоциативности и коммутативности этих операций вытекают из аналогичных свойств числового сложения и умножения. То же замечание относится к свойству дистрибутивности. Роль нулевого элемента при сложении играет класс $\overline{0}$. Противоположным элементом для класса вычетов $\overline{r} \neq \overline{0}$ является класс $\overline{n-r}$. Из определения сложения классов следует, что

$$\overline{r}+(\overline{n-r})=\overline{0}.$$

В общем определении кольца не содержится требование коммутативности умножения. В том случае, если умножение обладает этим дополнительным свойством, кольцо называется *коммутативным*. В примерах 1—3 мы имеем как раз коммутативные кольца, а позднее (в приложении 5) познакомимся с важным примером некоммутативного кольца.

Умножение в кольце, как и в группе, является ассоциативной операцией, но другие свойства группового умножения в кольце могут не выполняться. Правда, большинство важных колец содержат единичный элемент относительно умножения (скажем, кольца из примеров 1—3), но и такие кольца заведомо содержат элементы, для которых не существует обратного (необратимые элементы). В любом кольце необратим элемент 0 — нулевой элемент относительно сложения, так как доказывается, что он отличен от единичного и что для любого $a \in K$ имеет место:

$$0 \cdot a = a \cdot 0 = 0,$$

Как показывают примеры 1 и 2, необратимыми могут быть и ненулевые

элементы. Так, в кольце целых чисел обратимы лишь 1 и —1, всякое другое $n \neq 0$ в кольце $\mathbb{Z}$ не имеет обратного, так как $1/n \notin \mathbb{Z}$.

Рассмотрим таблицы умножения ненулевых элементов в кольцах вычетов $\mathbb{Z}_4$ и $\mathbb{Z}_5$.

Таблица 19

$\cdot$	$\bar{1}$	$\bar{2}$	$\bar{3}$
$\bar{1}$	$\bar{1}$	$\bar{2}$	$\bar{3}$
$\bar{2}$	$\bar{2}$	$\bar{0}$	$\bar{2}$
$\bar{3}$	$\bar{3}$	$\bar{2}$	$\bar{1}$

Таблица 20

$\cdot$	$\bar{1}$	$\bar{2}$	$\bar{3}$	$\bar{4}$
$\bar{1}$	$\bar{1}$	$\bar{2}$	$\bar{3}$	$\bar{4}$
$\bar{2}$	$\bar{2}$	$\bar{4}$	$\bar{1}$	$\bar{3}$
$\bar{3}$	$\bar{3}$	$\bar{1}$	$\bar{4}$	$\bar{2}$
$\bar{4}$	$\bar{4}$	$\bar{3}$	$\bar{2}$	$\bar{1}$

Таблица 19 показывает, что в кольце могут существовать ненулевые элементы, произведение которых равно нулю: в $\mathbb{Z}_4$ $\bar{2}\cdot\bar{2}=\bar{0}$. Из этой же таблицы видно, что класс $\bar{2}$ необратим. Вообще, можно доказать, что ненулевые элементы, произведение которых равно нулю (называемые *делителями нуля*), всегда необратимы. С другой стороны, таблица 20 показывает, что в кольце $\mathbb{Z}_5$ всякий ненулевой элемент обратим. Кольца с этим свойством имеют особое значение. Примем такое определение.

Коммутативное кольцо с единицей, в котором всякий ненулевой элемент обратим, называется *полем*.

Множество ненулевых элементов поля относительно умножения образует в силу определения поля коммутативную группу, которая называется *мультипликативной группой поля*.

Простейшими примерами числовых полей являются поле рациональных чисел $\mathbb{Q}$ и поле действительных чисел $\mathbb{R}$ (разумеется, относительно операций сложения и умножения чисел). Полем, как ясно из предыдущего, является и кольцо $\mathbb{Z}_5$. Вообще, можно доказать, что при любом простом p (и только в этом случае) кольцо вычетов $\mathbb{Z}_p$ является полем. Оно называется *полем вычетов по модулю p*. В таблице 21 указаны элементы, обратные к ненулевым элементам поля вычетов $\mathbb{Z}_7$.

Поля вычетов $\mathbb{Z}_p$ являются простейшими примерами конечных полей. В алгебре доказывается, что в произвольном конечном поле число элементов q всегда есть степень простого числа: $q=p^n$. Справедливо и обратное утверждение: для любого q, являющегося степенью простого числа, существует поле с q элементами.

Таблица 21

$\bar{k}$	$\bar{1}$	$\bar{2}$	$\bar{3}$	$\bar{4}$	$\bar{5}$	$\bar{6}$
$\bar{k}^{-1}$	$\bar{1}$	$\bar{4}$	$\bar{5}$	$\bar{2}$	$\bar{3}$	$\bar{6}$

Конечные поля часто называют полями Галуа (их обозначают $GF(q)$); важное их свойство, используемое, в частности, и в теории кодирования, состоит в следующем:

Мультипликативная группа поля Галуа является циклической группой порядка $q-1$.

Образующий элемент мультипликативной группы поля Галуа называют ***примитивным элементом***. Так, в поле $\mathbb{Z}_7$ примитивным элементом является класс вычетов $\bar{3}$. Действительно, его степени

$$\bar{3},\quad \bar{3}^2=\bar{2},\quad \bar{3}^3=\bar{6},\quad \bar{3}^4=\bar{4},\quad \bar{3}^5=\bar{5},\quad \bar{3}^6=\bar{1}$$

исчерпывают все ненулевые элементы поля.

Заметим, что класс $\bar{2}$ не является примитивным элементом в $\mathbb{Z}_7$, так как среди его степеней нет, например, класса $\bar{3}$. В то же время имеется очень много простых чисел p, для которых элемент $\bar{2}$ примитивен в $\mathbb{Z}_p$. Так обстоит дело в полях $\mathbb{Z}_3$, $\mathbb{Z}_5$, $\mathbb{Z}_{11}$ и т. д. В теории чисел известна следующая до сих пор не решенная задача:

Бесконечно ли множество тех простых чисел p, для которых $\bar{2}$ является примитивным элементом в $\mathbb{Z}_p$?

Интересно, что с ответом на этот вопрос связано решение некоторых проблем теории кодирования.

В заключение определим еще одно важное понятие, связанное с кольцом,— понятие идеала.

Подмножество I кольца K называется его (двусторонним) *идеалом*, если оно само является кольцом относительно операций на K и если для любых элементов $a\in K$ и $b\in I$ оба произведения ab и ba принадлежат I.

Так, множество четных чисел есть идеал кольца $\mathbb{Z}$. Читатель легко проверит, что и вообще всякое множество чисел, кратных какому-нибудь числу k, является идеалом кольца $\mathbb{Z}$.

Рекомендуем читателю найти идеалы колец вычетов $\mathbb{Z}_5$, $\mathbb{Z}_6$, $\mathbb{Z}_9$ и кольца многочленов $\mathbb{R}[X]$.

4. Арифметическое n-мерное векторное пространство

Всякая точка на плоскости при выбранной системе координат задается парой (α, β) своих координат; числа α и β можно понимать также как координаты радиуса-вектора с концом в этой точке. Аналогично, в пространстве тройка (α, β, γ) определяет точку или вектор с координатами α, β, γ. Именно на этом основывается хорошо известная читателю геометрическая интерпретация систем линейных уравнений с двумя или тремя неизвестными. Так, в случае системы двух линейных уравнений с двумя неизвестными

$$a_1x + b_1y = c_1,$$
$$a_2x + b_2y = c_2$$

каждое из уравнений истолковывается как прямая на плоскости (см. рис. 26), а решение (α, β) — как точка пересечения этих прямых или

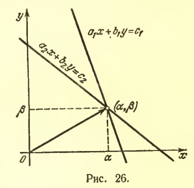


Рис. 26.

как вектор с координатами α и β (рисунок соответствует случаю, когда система имеет единственное решение).

Аналогично можно поступить с системой линейных уравнений с тремя неизвестными, интерпретируя каждое уравнение как уравнение плоскости в пространстве.

В математике и различных ее приложениях (в частности, в теории кодирования) приходится иметь дело с системами линейных уравнений, содержащих более трех неизвестных. Системой линейных уравнений с n неизвестными $x_1, x_2, \ldots, x_n$ называется совокупность уравнений вида

$$\begin{aligned} a_{11}x_1 + a_{12}x_2 + \ldots + a_{1n}x_n &= b_1, \\ a_{21}x_1 + a_{22}x_2 + \ldots + a_{2n}x_n &= b_2, \\ \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \\ a_{m1}x_1 + a_{m2}x_2 + \ldots + a_{mn}x_n &= b_m, \end{aligned} \qquad (1)$$

где a_{ij} и b_i — произвольные действительные числа. Число уравнений в системе может быть любым и никак не связано с числом неизвестных. Коэффициенты при неизвестных a_{ij} имеют двойную нумерацию: первый индекс i указывает номер уравнения, второй индекс j — номер неизвестного, при котором стоит данный коэффициент. Всякое решение системы понимается как набор (действительных) значений неизвестных $(\alpha_1, \alpha_2, \ldots, \alpha_n)$, обращающих каждое уравнение в верное равенство.

Хотя непосредственное геометрическое истолкование системы (1) при $n>3$ уже невозможно, однако вполне возможно и во многих отношениях удобно распространить на случай произвольного n геометрический язык пространства двух или трех измерений. Этой цели и служат дальнейшие определения.

Всякий упорядоченный набор из n действительных чисел $(\alpha_1, \alpha_2, \ldots, \alpha_n)$ называется *n-мерным арифметическим вектором*, а сами числа $\alpha_1, \alpha_2, \ldots, \alpha_n$ — *координатами* этого вектора.

Для обозначения векторов используется, как правило, жирный шрифт и для вектора $\mathbf{a}$ с координатами $\alpha_1, \alpha_2, \ldots, \alpha_n$ сохраняется обычная форма записи:

$$\mathbf{a}=(\alpha_1, \alpha_2, \ldots \alpha_n).$$

По аналогии с обычной плоскостью множество всех n-мерных векторов, удовлетворяющих линейному уравнению с n неизвестными, называют *гиперплоскостью* в n-мерном пространстве. При таком определении множество всех решений системы (1) есть не что иное, как пересечение нескольких гиперплоскостей.

Сложение и умножение n-мерных векторов определяются по тем же правилам, что и для обычных векторов. А именно, если

$$\mathbf{a}=(\alpha_1, \alpha_2, \ldots, \alpha_n), \quad \mathbf{b}=(\beta_1, \beta_2, \ldots, \beta_n) \tag{2}$$

— два n-мерных вектора, то их суммой называется вектор

$$\mathbf{a}+\mathbf{b}=(\alpha_1+\beta_1, \alpha_2+\beta_2, \ldots, \alpha_n+\beta_n). \tag{3}$$

Произведением вектора $\mathbf{a}$ на число λ называется вектор

$$\lambda\mathbf{a}=(\lambda\alpha_1, \lambda\alpha_2, \ldots, \lambda\alpha_n). \tag{4}$$

Множество всех n-мерных арифметических векторов с операциями сложения векторов и умножения вектора на число называется *арифметическим n-мерным векторным пространством L_n*.

Используя введенные операции, можно рассматривать произвольные линейные комбинации нескольких векторов, т. е. выражения вида

$$\lambda_1\mathbf{a}_1+\lambda_2\mathbf{a}_2+\ldots+\lambda_k\mathbf{a}_k,$$

где λ_i — действительные числа. Например, линейная комбинация векторов (2) с коэффициентами λ и μ — это вектор

$$\lambda\mathbf{a}+\mu\mathbf{b}=(\lambda\alpha_1+\mu\beta_1, \lambda\alpha_2+\mu\beta_2, \ldots, \lambda\alpha_n+\mu\beta_n).$$

В трехмерном пространстве векторов особую роль играет тройка векторов $\boldsymbol{i}$, $\boldsymbol{j}$, $\boldsymbol{k}$ (координатные орты), по которым разлагается любой вектор $\boldsymbol{a}$:

$$\boldsymbol{a} = x\boldsymbol{i} + y\boldsymbol{j} + z\boldsymbol{k},$$

где x, y, z — действительные числа (координаты вектора $\boldsymbol{a}$).

В n-мерном случае такую же роль играет следующая система векторов:

$$\begin{aligned} \boldsymbol{e}_1 &= (1, 0, 0, \ldots, 0), \\ \boldsymbol{e}_2 &= (0, 1, 0, \ldots, 0), \\ \boldsymbol{e}_3 &= (0, 0, 1, \ldots, 0), \\ &\ldots\ldots\ldots\ldots \\ \boldsymbol{e}_n &= (0, 0, 0, \ldots, 1). \end{aligned} \tag{5}$$

Всякий вектор $\boldsymbol{a}$ есть, очевидно, линейная комбинация векторов $\boldsymbol{e}_1$, $\boldsymbol{e}_2$, ..., $\boldsymbol{e}_n$:

$$\boldsymbol{a} = \alpha_1\boldsymbol{e}_1 + \alpha_2\boldsymbol{e}_2 + \ldots + \alpha_n\boldsymbol{e}_n, \tag{6}$$

причем коэффициенты α_1, α_2, ..., α_n совпадают с координатами вектора $\boldsymbol{a}$.

Обозначая через $\boldsymbol{0}$ вектор, все координаты которого равны нулю (кратко, *нулевой вектор*), введем следующее важное определение:

Система векторов $\boldsymbol{a}_1$, $\boldsymbol{a}_2$, ..., $\boldsymbol{a}_k$ называется *линейно зависимой*, если существует равная нулевому вектору линейная комбинация

$$\lambda_1\boldsymbol{a}_1 + \lambda_2\boldsymbol{a}_2 + \ldots + \lambda_k\boldsymbol{a}_k = \boldsymbol{0},$$

в которой хотя бы один из коэффициентов λ_1, λ_2, ..., λ_k отличен от нуля. В противном случае система называется *линейно независимой*.

Так, векторы

$$\boldsymbol{a}_1 = (1, 0, 1, 1), \ \boldsymbol{a}_2 = (1, 2, 1, 1), \ \boldsymbol{a}_3 = (2, 2, 2, 2)$$

линейно зависимы, поскольку

$$\boldsymbol{a}_1 + \boldsymbol{a}_2 - \boldsymbol{a}_3 = \boldsymbol{0}.$$

Линейная зависимость, как видно из определения, равносильна (при $k \geqslant 2$) тому, что хотя бы один из векторов системы является линейной комбинацией остальных.

Если система состоит из двух векторов $\boldsymbol{a}_1$, $\boldsymbol{a}_2$, то линейная зависимость системы означает, что один из векторов пропорционален другому, скажем, $\boldsymbol{a}_1 = \lambda\boldsymbol{a}_2$; в трехмерном случае это равносильно *коллинеарности* векторов $\boldsymbol{a}_1$ и $\boldsymbol{a}_2$. Точно так же линейная зависимость системы из трех векторов в обычном пространстве означает *компланарность* этих векторов. Понятие линейной зависимости является, таким образом, естественным обобщением понятий коллинеарности и компланарности.

Нетрудно убедиться, что векторы $e_1, e_2, \ldots, e_n$ из системы (5) линейно независимы. Следовательно, в n-мерном пространстве существуют системы из n линейно независимых векторов. Можно показать, что всякая система из большего числа векторов линейно зависима.

Всякая система $a_1, a_2, \ldots, a_n$ из n линейно независимых векторов n-мерного пространства L_n называется его *базисом*.

Любой вектор a пространства L_n раскладывается, и притом единственным образом, по векторам произвольного базиса $a_1, a_2, \ldots, a_n$:

$$a = \lambda_1 a_1 + \lambda_2 a_2 + \ldots + \lambda_n a_n.$$

Этот факт легко устанавливается на основании определения базиса.

Продолжая аналогию с трехмерным пространством, можно и в n-мерном случае определить *скалярное произведение* $a \cdot b$ векторов, полагая

$$a \cdot b = \alpha_1\beta_1 + \alpha_2\beta_2 + \ldots + \alpha_n\beta_n.$$

При таком определении сохраняются все основные свойства скалярного произведения трехмерных векторов. Векторы a и b называются *ортогональными*, если их скалярное произведение равно нулю:

$$\alpha_1\beta_1 + \alpha_2\beta_2 + \ldots + \alpha_n\beta_n = 0.$$

В теории линейных кодов используется еще одно важное понятие — понятие подпространства. Подмножество V пространства L_n называется подпространством этого пространства, если

1) для любых векторов a, b, принадлежащих V, их сумма $a+b$ также принадлежит V;

2) для любого вектора a, принадлежащего V, и для любого действительного числа λ вектор λa также принадлежит V.

Например, множество всех линейных комбинаций векторов e_1, e_2 из системы (5) будет подпространством пространства L_n.

В линейной алгебре доказывается, что во всяком подпространстве V существует такая линейно независимая система векторов $a_1, a_2, \ldots, a_k$, что всякий вектор a подпространства является линейной комбинацией этих векторов:

$$a = \lambda_1 a_1 + \lambda_2 a_2 + \ldots + \lambda_k a_k.$$

Указанная система векторов называется базисом подпространства V.

Из определения пространства и подпространства непосредственно следует, что пространство L_n есть коммутативная группа относительно операции сложения векторов, а любое его подпространство V является подгруппой этой группы. В этом смысле можно, например, рассматривать смежные классы пространства L_n по подпространству V.

В заключение подчеркнем, что если в теории n-мерного арифметического пространства вместо действительных чисел (т. е. элементов

поля действительных чисел) рассматривать элементы произвольного поля F, то все определения и факты, приведенные выше, сохранили бы силу.

В теории кодирования важную роль играет случай, когда поле F есть поле вычетов $\mathbb{Z}_p$, которое, как мы знаем, конечно. В этом случае соответствующее n-мерное пространство также конечно и содержит, как нетрудно видеть, p^n элементов.

Понятие пространства, как и понятия группы и кольца, допускает также и аксиоматическое определение. За подробностями мы отсылаем читателя к любому курсу линейной алгебры.

5. Алгебра матриц

Большую роль в линейной алгебре и многих других областях математики играют так называемые матрицы.

Под *матрицей* понимают прямоугольную таблицу вида

$$\begin{pmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ \cdot & \cdot & \cdot & \cdot \\ a_{m1} & a_{m2} & \dots & a_{mn} \end{pmatrix}, \qquad (1)$$

где буквы a_{ij} обозначают некоторые элементы. Эти обозначения содержат два индекса, первый из которых указывает номер строки, а второй — номер столбца, на пересечении которых находится соответствующий элемент. Про матрицу, у которой m строк и n столбцов, говорят, что она имеет порядок $m \times n$. В случае, когда число строк и столбцов одинаково ($m=n$), матрица называется *квадратной* порядка n.

Если $m=1$, то матрицу можно понимать как вектор, координаты которого записаны в строку; ее тогда так и называют *вектор-строка*. Точно так же в случае одностолбцовой матрицы пользуются термином *вектор-столбец*. Иногда матрицу обозначают одной буквой (A, B и т. д.), а вместо (1) часто используется сокращенное обозначение (a_{ij}).

Если в исходной матрице A строки и столбцы поменять ролями, то получим матрицу, называемую транспонированной к A (обозначается A^T). Матрица, транспонированная к матрице (1), имеет вид:

$$\begin{pmatrix} a_{11} & a_{21} & \dots & a_{m1} \\ a_{12} & a_{22} & \dots & a_{m2} \\ \cdot & \cdot & \cdot & \cdot \\ a_{1n} & a_{2n} & \dots & a_{mn} \end{pmatrix}.$$

Для матриц с действительными элементами определяются операции сложения матриц и умножения матрицы на число, аналогичные операциям над векторами, а именно, для любых двух матриц порядка $m \times n$

$$\begin{pmatrix} a_{11} & \dots & a_{1n} \\ \cdot & \cdot & \cdot \\ a_{m1} & \dots & a_{mn} \end{pmatrix} + \begin{pmatrix} b_{11} & \dots & b_{1n} \\ \cdot & \cdot & \cdot \\ b_{m1} & \dots & b_{mn} \end{pmatrix} = \begin{pmatrix} a_{11}+b_{11} & \dots & a_{1n}+b_{1n} \\ \cdot & \cdot & \cdot \\ a_{m1}+b_{m1} & \dots & a_{mn}+b_{mn} \end{pmatrix}$$

и для любой матрицы и любого числа α

$$\alpha\begin{pmatrix} a_{11} & \dots & a_{1n} \\ \cdot & \cdot\cdot\cdot & \cdot \\ a_{m1} & \dots & a_{mn} \end{pmatrix}=\begin{pmatrix} \alpha a_{11} & \dots & \alpha a_{1n} \\ \cdot & \cdot\cdot\cdot & \cdot \\ \alpha a_{m1} & \dots & \alpha a_{mn} \end{pmatrix}.$$

Определение этих операций выглядит вполне естественно. Иначе обстоит дело с довольно своеобразной операцией умножения матриц. Рассмотрим сначала умножение квадратных матриц одного порядка n. Если

$$A=(a_{ij}) \quad \text{и} \quad B=(b_{ij})$$

— две такие матрицы, то их произведением

$$C=A\cdot B=(c_{ij})$$

называется квадратная матрица порядка n, произвольный элемент которой вычисляется по правилу:

$$c_{ij}=a_{i1}b_{1j}+a_{i2}b_{2j}+\dots+a_{in}b_{nj}. \qquad (2)$$

Иначе говоря, чтобы получить элемент i-й строки и j-го столбца матрицы $C=A\cdot B$, нужно взять сумму произведений элементов i-й строки матрицы A на соответствующие элементы j-го столбца матрицы B. Например,

$$\begin{pmatrix} 1 & 2 \\ 0 & 1 \end{pmatrix}\begin{pmatrix} 2 & 0 \\ -1 & 1 \end{pmatrix}=\begin{pmatrix} 0 & 2 \\ -1 & 1 \end{pmatrix}.$$

Умножение матриц некоммутативно, т. е., вообще говоря, $A\cdot B\neq B\cdot A$. Так, перемножив две предыдущие матрицы в обратном порядке, получим иную матрицу:

$$\begin{pmatrix} 2 & 0 \\ -1 & 1 \end{pmatrix}\cdot\begin{pmatrix} 1 & 2 \\ 0 & 1 \end{pmatrix}=\begin{pmatrix} 2 & 4 \\ -1 & -1 \end{pmatrix}.$$

Вместе с тем, нетрудно доказать, что умножение матриц ассоциативно:

$$(A\cdot B)\cdot C=A\cdot(B\cdot C).$$

Особую роль (подобную числовой единице) играет так называемая *единичная матрица*

$$E=\begin{pmatrix} 1 & 0 & & 0 \\ 0 & 1 & & \\ & & \ddots & \\ 0 & 0 & & 1 \end{pmatrix}.$$

Действительно, из формулы (2) следует, что

$$EA=AE=A$$

для любой квадратной матрицы A.

Можно убедиться, что множество всех квадратных матриц заданного порядка образует относительно введенных операций сложения и умножения (некоммутативное) кольцо.

Правило умножения матриц можно распространить и на прямоугольные матрицы, не являющиеся квадратными. Формула (2) позволяет это сделать, если число столбцов первой матрицы A совпадает с числом строк матрицы B. Например,

$$\begin{pmatrix} 1 & 2 \\ 3 & 1 \\ 0 & -1 \end{pmatrix} \cdot \begin{pmatrix} 2 & 0 \\ -1 & 1 \end{pmatrix} = \begin{pmatrix} 0 & 2 \\ 5 & 1 \\ 1 & -1 \end{pmatrix}.$$

При этом получающаяся матрица имеет столько же строк, сколько первый сомножитель, и столько же столбцов, сколько второй. Пользуясь операциями над матрицами, мы получаем возможность записывать произвольные системы уравнений в краткой матричной форме. Действительно, пусть матрица

$$A = \begin{pmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ \dots & \dots & \dots & \dots \\ a_{m1} & a_{m2} & \dots & a_{mn} \end{pmatrix}$$

составлена из коэффициентов при неизвестных системы (1) приложения 4 (в этом случае она называется матрицей системы). Рассмотрим векторы-столбцы неизвестных и свободных членов, обозначая их соответственно через $\boldsymbol{x}$ и $\boldsymbol{b}$:

$$\boldsymbol{x} = \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix}, \quad \boldsymbol{b} = \begin{pmatrix} b_1 \\ b_2 \\ \vdots \\ b_m \end{pmatrix}.$$

Тогда произведение $A\boldsymbol{x}$ есть матрица с m строками и одним столбцом, т. е. вектор-столбец, элементы которого вычисляются согласно формуле (2). Таким образом,

$$A\boldsymbol{x} = \begin{pmatrix} a_{11}x_1 + a_{12}x_2 + \dots + a_{1n}x_n \\ a_{21}x_1 + a_{22}x_2 + \dots + a_{2n}x_n \\ \dots \dots \dots \dots \\ a_{m1}x_1 + a_{m2}x_2 + \dots + a_{mn}x_n \end{pmatrix}.$$

Каждое из уравнений системы (1) означает равенство соответствующих координат вектора $A\boldsymbol{x}$ и вектора $\boldsymbol{b}$, и вся система в целом означает тогда равенство

$$A\boldsymbol{x} = \boldsymbol{b}.$$

Полученная краткая запись и есть матричная форма системы линейных уравнений. Подобная матричная запись встречается во множестве других ситуаций, и она широко используется в математической, физической и технической литературе.

Все сказанное о матрицах с действительными элементами в равной мере относится к матрицам с элементами из произвольного поля F. Так, в теории кодирования приходится рассматривать матрицы, составленные из элементов конечного поля. Естественно, что при оперировании с такими матрицами их элементы складываются и умножаются в соответствии с «арифметикой» поля F. Вот, к примеру, как перемножаются две матрицы с элементами из $\mathbb{Z}_2$:

$$\begin{pmatrix} 1 & 0 & 0 & 1 \\ 0 & 1 & 1 & 0 \\ 1 & 1 & 1 & 0 \end{pmatrix} \cdot \begin{pmatrix} 1 & 0 \\ 1 & 1 \\ 0 & 1 \\ 1 & 1 \end{pmatrix} = \begin{pmatrix} 0 & 1 \\ 1 & 0 \\ 0 & 0 \end{pmatrix}.$$

6. Задачи и дополнения

1. Покажите, что преобразование, обратное параллельному переносу в направлении вектора a, есть параллельный перенос в противоположном направлении (на вектор $-a$).

2. Пусть f — симметрия плоскости относительно оси Ox, g — поворот плоскости на $\pi/2$ против часовой стрелки. Проверьте, что преобразование fg есть симметрия относительно биссектрисы 1-го и 3-го координатных углов. Найдите также преобразование gf и убедитесь, что $fg \neq gf$.

3. Операция произведения преобразований обладает свойством ассоциативности, это значит, что для любой тройки преобразований f_1, f_2, f_3 выполняется равенство

$$(f_1 f_2) f_3 = f_1 (f_2 f_3).$$

Справедливость соотношения вытекает из того, что обе его части есть результат последовательного выполнения трех преобразований сначала f_1, затем f_2, затем f_3.

4. Какие из следующих множеств образуют группу преобразований плоскости:

а) множество всех параллельных переносов;

б) множество всех вращений относительно фиксированного центра;

в) множество всех вращений плоскости;

г) множество всевозможных осевых симметрий.

5. Напомним, что совокупность всех подстановок n-элементного множества образует группу. Она называется симметрической группой степени n и обозначается S_n. Найдите все подстановки из S_3, не меняющие значения функций

а) $\varphi_1 = x_1x_2 + x_2x_3 + x_1x_3$;

б) $\varphi_2 = (x_1 - x_2)(x_1 - x_3)(x_2 - x_3)$.

Проверьте прямым вычислением, что найденные в каждом из случаев подстановки образуют группы.

6. *Транспозицией* называется подстановка, переставляющая какие-либо два символа, а все прочие оставляющая на месте. Например, из двух подстановок

$$\begin{pmatrix} 1 & 2 & 3 & 4 \\ 1 & 3 & 2 & 4 \end{pmatrix}, \quad \begin{pmatrix} 1 & 2 & 3 & 4 \\ 2 & 3 & 4 & 1 \end{pmatrix}$$

первая является транспозицией, вторая — нет, но она может быть представлена в виде произведения транспозиций следующим образом:

$$\begin{pmatrix} 1 & 2 & 3 & 4 \\ 2 & 3 & 4 & 1 \end{pmatrix} = \begin{pmatrix} 1 & 2 & 3 & 4 \\ 2 & 1 & 3 & 4 \end{pmatrix} \begin{pmatrix} 1 & 2 & 3 & 4 \\ 3 & 2 & 1 & 4 \end{pmatrix} \begin{pmatrix} 1 & 2 & 3 & 4 \\ 4 & 2 & 3 & 1 \end{pmatrix}. \qquad (1)$$

Справедлив общий факт: всякая подстановка разлагается в произведение транспозиций.

Это разложение неоднозначно, например, вместо (1) мы могли бы написать

$$\begin{pmatrix} 1 & 2 & 3 & 4 \\ 2 & 3 & 4 & 1 \end{pmatrix} = \begin{pmatrix} 1 & 2 & 3 & 4 \\ 1 & 3 & 2 & 4 \end{pmatrix} \begin{pmatrix} 1 & 2 & 3 & 4 \\ 1 & 4 & 3 & 2 \end{pmatrix} \begin{pmatrix} 1 & 2 & 3 & 4 \\ 2 & 1 & 3 & 4 \end{pmatrix}.$$

Однако можно доказать, что при всех представлениях подстановки в виде произведения транспозиций число сомножителей имеет одинаковую четность.

Подстановка называется четной, если число транспозиций и ее разложении четно, в противном случае подстановка называется нечетной.

7. Покажите, что множество всех четных подстановок образует группу, она обозначается A_n и называется *знакопеременной группой*.

8. Убедитесь, что множество подстановок из задачи 5 (*б*) совпадает со знакопеременной группой A_3.

Вообще рассмотрим функцию

$$F(x_1, x_2, \ldots, x_n) = \prod_{i<j} (x_i - x_j), \qquad i, j = 1, 2, \ldots, n. \qquad (2)$$

Тогда множество всех подстановок переменных, не меняющих значения этой функции, совпадает со знакопеременной группой A_n. Это вытекает из того, что при любой транспозиции значение функции (2) меняется на противоположное, например,

$$F(x_2, x_1, \ldots, x_n) = -F(x_1, x_2, \ldots, x_n).$$

Сказанное объясняет и происхождение термина «знакопеременная группа».

9. Для числовых групп типичной является такая ситуация, когда все степени одного элемента $g \neq 1$ различны *). Если же обратиться к группам преобразований, то в них зачастую бывает так, что две различные степени одного элемента (преобразования) совпадают. В качестве примера таких элементов рассмотрите преобразования: а) симметрию относительно оси; б) поворот плоскости на угол $\pi/3$ (вообще на угол $2\pi/k$).

Что касается конечных групп, то здесь для любого элемента g существуют такие натуральные k и m $(k \neq m)$, что

$$g^k = g^m. \tag{3}$$

Если бы это было не так, то группа содержала бы бесконечно много элементов.

Пусть в равенстве (3) $k > m$. Умножим обе его части на элемент g^{-m}, это даст следующее равенство $g^{k-m} = e$, или $g^n = e$, где $n = k - m > 0$.

В описанной ситуации обязательно найдется наименьшее натуральное n со свойством $g^n = e$, которое называется *порядком* элемента g.

Если же все степени элемента g

$$g^0 = e,\ g,\ g^2,\ \ldots,\ g^k$$

различны, то g называется элементом бесконечного порядка.

10. Найдите порядки следующих подстановок:

$$\begin{pmatrix} 1 & 2 & 3 & 4 & 5 \\ 2 & 3 & 4 & 5 & 1 \end{pmatrix}, \quad \begin{pmatrix} 1 & 2 & 3 & 4 & 5 \\ 2 & 3 & 1 & 5 & 4 \end{pmatrix}, \quad \begin{pmatrix} 1 & 2 & 3 & 4 & 5 \\ 1 & 4 & 5 & 2 & 3 \end{pmatrix}.$$

11. Пусть $G = (g)$ — конечная циклическая группа, и ее образующий g имеет порядок n. Рассмотрим произвольную степень g^m.

Если показатель кратен n $(m = sn)$, то $g^m = g^{ns} = (g^n)^s = e^s = e$. Если же показатель m произволен, то его всегда можно записать в виде: $m = sn + r$, $0 \leqslant r \leqslant n - 1$, где r — остаток от деления m на n. Но тогда $g^m = g^{sn+r} = g^{sn} \cdot g^r = e \cdot g^r = g^r$. Это значит, что все элементы группы $G = (g)$ исчерпываются следующими n элементами:

$$e,\ g,\ g^2,\ \ldots,\ g^{n-1}.$$

Мы доказали, что порядок конечной циклической группы совпадает с порядком ее образующего. Как, используя сказанное, вывести из теоремы Лагранжа следующие утверждения:

а) порядок любого элемента конечной группы является делителем порядка этой группы;

б) всякая группа простого порядка является циклической?

*) В случае, если групповая операция — сложение, приходится говорить не о степенях, а о кратных элемента.

12. Найдите возможные порядки элементов в группах S_3, S_5, циклической группе порядка 12.

13. Проверьте, что всякая подгруппа H циклической группы $G=(g)$ сама является циклической.

Указание: рассмотрите наименьшую положительную степень $g^k \in H$, покажите, что $H=(g^k)$.

14. Покажем, пользуясь законом дистрибутивности, что нулевой элемент аддитивной группы кольца играет роль нуля и при умножении.

Рассмотрим произведение любых двух элементов a и b и преобразуем его следующим образом:

$$ab=(a+0)\,b=ab+0b.$$

Второе слагаемое, как видно из равенства, играет роль нейтрального элемента при сложении. В силу его единственности $0b=0$ для всякого $b \in K$.

15. Какие из следующих числовых множеств являются кольцами относительно обычных операций сложения и умножения:

а) множество четных чисел;

б) множество чисел, кратных данному n;

в) множество многочленов степени $\leqslant n$ с действительными коэффициентами;

г) множество всех многочленов с целыми коэффициентами;

д) множество неотрицательных действительных чисел;

е) множество всех чисел вида $a+b\sqrt{2}$, где a и b—рациональные.

В каких случаях мы имеем дело с кольцом с единицей, в каких — с полем?

16. Покажем, что делитель нуля (левый или правый) не может быть обратим. В самом деле, пусть $xa=1$ и $ab=0$. Тогда

$$xab=\begin{cases}(xa)\cdot b=1\cdot b=b,\\ x\,(ab)=x\cdot 0=0.\end{cases}$$

Следовательно, $b=0$ и a не является делителем нуля.

Из сказанного непосредственно вытекает, что никакое поле не содержит делителей нуля.

17. Показать, что если $n=n_1\cdot n_2$—число составное, то $\mathbb{Z}_n$ не является полем.

18. Показать, что ненулевой элемент $\overline{k} \in \mathbb{Z}_n$ обратим тогда и только тогда, когда числа k и n взаимно просты, и что в противном случае $\overline{k}$ является делителем нуля.

19. Пользуясь предыдущим утверждением, найдите обратные элементы и делители нуля в кольцах вычетов $\mathbb{Z}_6$, $\mathbb{Z}_8$, $\mathbb{Z}_{11}$, $\mathbb{Z}_{12}$. Для обратных элементов найдите обратные. Для элементов $\overline{k}_i$ явля-

ющихся делителями нуля, найдите $\overline{m} \neq 0$ такой, что $\overline{k} \cdot \overline{m} = \overline{0}$. Является ли такой класс $\overline{m}$ единственным?

20. Уже самые простые сведения о кольцах и группах могут быть применены в различных математических вопросах. Прекрасной иллюстрацией этого является теория чисел. Рассмотрим, например, хорошо известную теорему (малую теорему Ферма):

Если p — простое, то для любого натурального a число $a^p - a$ делится на p.

Рассуждаем так:

$$a^p - a = a(a^{p-1} - 1).$$

Если a делится на p, то утверждение очевидно.

Пусть a не делится на p и $a = mp + r$ $(0 < r \leqslant p - 1)$. Тогда в поле вычетов $\mathbb{Z}_p$ $a \in \overline{r}$ и $\overline{r} \neq \overline{0}$. Делимость числа $a^{p-1} - 1$ на p равносильна тому, что в $\mathbb{Z}_p$ справедливо равенство $\overline{r}^{p-1} - \overline{1} = \overline{0}$ или равносильное ему

$$\overline{r}^{p-1} = \overline{1}. \tag{4}$$

Так как порядок мультипликативной группы поля $\mathbb{Z}_p$ равен $p - 1$, то из теоремы Лагранжа вытекает справедливость равенства (4), и это доказывает малую теорему Ферма.

21. Подумайте (это не простая задача), как с помощью указанных методов можно доказать утверждение: если p — простое, то $(p-1)! + 1$ делится на p. Это теорема Вильсона, одна из наиболее изящных теорем элементарной теории чисел.

22. *Функция Эйлера* $\varphi(n)$ определяется в теории чисел как количество натуральных чисел, меньших n и взаимно простых с ним. Например, $\varphi(4) = 2$, $\varphi(8) = 4$ и т. д. Для всякого простого p очевидно $\varphi(p) = p - 1$. Обобщением малой теоремы Ферма является теорема Эйлера, которая формулируется так:

Для любого натурального a, взаимно простого с n, $a^{\varphi(n)} - 1$ делится на n. Например, при $n = 8$, $a = 5$ имеем $5^4 - 1 = 624 = 8 \times 78$. Можно предложить следующий план доказательства теоремы Эйлера (детали — на усмотрение читателя):

а) показать, что множество обратимых элементов любого кольца образует группу относительно умножения;

б) рассмотреть кольцо вычетов $\mathbb{Z}_n$ и убедиться, что группа его обратимых элементов имеет порядок $\varphi(n)$ (вспомните утверждение пункта 18);

в) дальнейшие рассуждения таковы же, как и при доказательстве малой теоремы Ферма.

23. Найти все решения системы линейных уравнений

$$x + 2z = 1, \quad y + 2z = 2, \quad 2x + z = 1$$

в поле вычетов: а) по модулю 3; б) по модулю 5.

24. Многочлен $p(X) \in F[X]$ называется неприводимым, если не существует разложения $p(X) = f(X)\,g(X)$, в котором $f(X), g(X) \in F[X]$ и степени сомножителей меньше, чем степень многочлена $p(X)$.

Выяснить, являются ли следующие многочлены:

$$X^3 + X^2 + 1, \quad X^4 + X^2 + 1, \quad X^2 - 2$$

неприводимыми над полем: а) $\mathbb{Z}_2$; б) $\mathbb{Z}_3$; в) $\mathbb{Q}$; г) $\mathbb{R}$.

25. Разложить на неприводимые множители многочлен

$$X^4 + X^3 + X + 1$$

над полем: а) $\mathbb{Z}_2$; б) $\mathbb{Z}_3$; в) $\mathbb{Q}$.

26. Найти наибольший общий делитель многочленов

$$f(X) = X^3 + X^2 + 2X + 2; \quad g(X) = X^2 + X + 1;$$

над полем: а) $\mathbb{Z}_3$; б) $\mathbb{Q}$.

27. Доказать что в конечном кольце с единицей любой ненулевой элемент либо обратим, либо является делителем нуля. Верно ли это утверждение без предположения конечности (вспомните кольцо $\mathbb{Z}$)?

28. Используя предыдущее утверждение, докажите, что конечное коммутативное кольцо без делителей нуля, содержащее более одного элемента, является полем.

29. В вопросах, связанных с решением систем линейных уравнений, особую роль играет понятие определителя квадратной матрицы. Введем здесь это понятие применительно к матрице 2-го порядка

$$\begin{pmatrix} \alpha & \beta \\ \gamma & \delta \end{pmatrix} \tag{5}$$

с элементами из произвольного поля F. Ее определителем называется величина

$$\Delta = \alpha\delta - \beta\gamma. \tag{6}$$

Доказать, что в кольце всех матриц 2-го порядка матрица (5) обратима тогда и только тогда, когда ее определитель отличен от нуля (указать способ отыскания обратной матрицы). В противном случае матрица (5) является делителем нуля.

30. Какие из перечисленных ниже матриц обратимы или являются делителями нуля

$$\begin{pmatrix} 1 & 2 \\ 2 & 1 \end{pmatrix}; \quad \begin{pmatrix} 1 & 0 \\ 0 & 2 \end{pmatrix}; \quad \begin{pmatrix} 1 & 1 \\ 2 & 2 \end{pmatrix}$$

над полем: а) $\mathbb{Z}_3$; б) $\mathbb{Q}$?

В случае, если матрица обратима, найти обратную.

ЛИТЕРАТУРА

1. *Шеннон К.* Работы по теории информации и кибернетике.— М.: ИЛ, 1963.
2. *Яглом А. М., Яглом И. М.* Вероятность и информация.— М.: Наука, 1973.
3. *Бриллюэн Л.* Наука и теория информации.— М.: Физматгиз, 1959.
4. *Холл М.* Комбинаторика.— М.: Мир, 1970.
5. *Оре О.* Приглашение в теорию чисел.— М.: Наука, 1980.
6. *Реньи А.* Трилогия о математике.— М.: Мир, 1980.
7. *Кострикин А. И.* Введение в алгебру.— М.: Наука, 1977.
8. *Калужнин Л. А.* Введение в общую алгебру.— М.: Наука, 1973.
9. *Фано Р.* Передача информации. Статистическая теория связи.— М.: Мир, 1965.
10. *Галлагер Р.* Теория информации и надежная связь.— М.: Советское радио, 1974.
11. *Питерсон У., Уэлдон Э.* Коды, исправляющие ошибки.— М.: Мир, 1976.
12. *Мак-Вильямс Ф., Слоэн Н. Дж.* Теория кодов, исправляющих ошибки.— М.: Связь, 1979.
13. *Касами Т., Токура Н., Ивадари Е., Инагаки Я.* Теория кодирования.— М.: Мир, 1978.
14. *Берлекэмп Э.* Алгебраическая теория кодирования.—М.: Мир, 1971.
15. *Колесник В. Д., Мирончиков Е. Т.* Декодирование циклических кодов.— М.: Связь, 1968.
16. *Стиффлер Дж.* Теория синхронной связи.— М. : Связь, 1975.
17. *Яблонский С. В.* Введение в дискретную математику.— М.: Наука, 1979.
18. *Блох Э. Л., Зяблов В. В.* Обобщенные каскадные коды.— М.: Связь, 1976.
19. *Марков А. А.* Введение в теорию кодирования.— М.: Наука, 1982.
20. *Новик Д. А.* Эффективное кодирование.— М. Л.: Энергия, 1965.
21. *Воробьев Н. Н.* Числа Фибоначчи.— М.: Наука, 1972.

СОДЕРЖАНИЕ

Михаил Наумович Аршинов
Леонид Ефимович Садовский
КОДЫ И МАТЕМАТИКА
(рассказы о кодировании)

(Серия: Библиотечка «Квант»)

Редактор *Ф. Л. Варпаховский*
Технический редактор *Е. В. Морозова*
Корректор *Е. В. Сидоркина*

ИБ № 12096

Сдано в набор 13. 12. 82. Подписано к печати 04. 04. 83. Т-08318. Бумага 84×108[1]/32. Бумага тип. № 3. Литературная гарнитура. Высокая печать. Условн. печ. л. 7,56. Уч.-изд. л. 8,21. Тираж 150 000 экз. Заказ № 1067. Цена 25 коп.

Издательство «Наука»
Главная редакция физико-математической литературы
117071, Москва, В-71, Ленинский проспект, 15

Ордена Октябрьской Революции и ордена Трудового Красного Знамени Первая Образцовая типография имени А. А. Жданова Союзполиграфпрома при Государственном комитете СССР по делам издательств, полиграфии и книжной торговли. Москва, М-54, Валовая, 28

Отпечатано в типографии № 2 изд-ва «Наука», Москва, Шубинский пер., 10. 2688

25 коп.

10

БИБЛИОТЕЧКА «КВАНТ»

ВЫШЛИ ИЗ ПЕЧАТИ:

Вып. 1. М. П. Бронштейн. Атомы и электроны.
Вып. 2. М. Фарадей. История свечи.
Вып. 3. О. Оре. Приглашение в теорию чисел.
Вып. 4. Опыты в домашней лаборатории.
Вып. 5. И. Ш. Слободецкий, Л. Г. Асламазов. Задачи по физике.
Вып. 6. Л. П. Мочалов. Головоломки.
Вып. 7. П. С. Александров. Введение в теорию групп.
Вып. 8. Г. Штейнгауз. Математический калейдоскоп.
Вып. 9. Замечательные ученые.
Вып. 10. В. М. Глушков, В. Я. Валах. Что такое ОГАС?
Вып. 11. Г. И. Копылов. Всего лишь кинематика.
Вып. 12. Я. А. Смородинский. Температура.
Вып. 13. А. Е. Карпов, Е. Я. Гик. Шахматный калейдоскоп.
Вып. 14. С. Г. Гиндикин. Рассказы о физиках и математиках.
Вып. 15. А. А. Боровой. Как регистрируют частицы.
Вып. 16. М. И. Каганов, В. М. Цукерник. Природа магнетизма.
Вып. 17. И. Ф. Шарыгин. Задачи по геометрии: Планиметрия.
Вып. 18. Л. В. Тарасов, А. Н. Тарасова. Беседы о преломлении света.
Вып. 19. А. Л. Эфрос. Физика и геометрия беспорядка.
Вып. 20. С. А. Пикин, Л. М. Блинов. Жидкие кристаллы.
Вып. 21. В. Г. Болтянский, В. А. Ефремович. Наглядная топология.
Вып. 22. М. И. Башмаков, Б. М. Беккер, В. М. Гольховой. Задачи по математике: Алгебра и анализ.
Вып. 23. А. Н. Колмогоров, И. Г. Журбенко, А. В. Прохоров. Введение в теорию вероятностей.
Вып. 24. Е. Я. Гик. Шахматы и математика.
Вып. 25. М. Д. Франк-Каменецкий. Самая главная молекула.
Вып. 26. В. С. Эдельман. Вблизи абсолютного нуля.
Вып. 27. С. Р. Филонович. Самая большая скорость.
Вып. 28. Б. С. Бокштейн. Атомы блуждают по кристаллу.
Вып. 29. А. В. Бялко. Наша планета — Земля.
Вып. 30. М. Н. Аршинов, Л. Е. Садовский. Коды и математика.